InforMik

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA"



Nasz komentarz

NABURMUSZENI I GNIEWNI

Rewolucja technologiczna sprawiła oto, że za sprawą komputera osobistego informatyka nagle przestała być prywatną sprawą profesjonalnych informatyków. Informatyka uważana była do niedawna — nie bez racji — za dziedzinę szczególnie trudną i wymagającą od swych kapłanów szczególnych predyspozycji intelektualnych. Przeciętny człowiek nie próbował nawet wnikać w magiczny świat „mózgów elektronowych", jak do niedawna zwykli nazywać komputery żurnaliści gazet popołudniowych, z góry uznając swą niekompetencję.

Dzisiaj sytuacja jakby odwróciła się: na informatyce znają się wszyscy, którzy przynajmniej raz w życiu dotknęli komputera. Długoletnie zaniedbania w systematycznej edukacji informatycznej społeczeństwa stworzyły w ludzkiej świadomości pustkę, wypełniającą się dziś pospiesznie, pod ciśnieniem fali mikrokomputerów, dość przypadkowymi treściami. Źródłem komputerowej wiedzy są najczęściej rubryki komputerowe w różnych gazetach i czasopismach, oraz wyspecjalizowane czasopisma, jak „Komputer" czy „Bajtek". Swój udział mają też audycje radiowe i telewizyjne.

Uderzeniowa dawka mikrokomputerów, głównie zresztą domowych, sprawiła, że nurt życia informatycznego rozdwoił się i popłynął dwoma oddzielnymi kanałami. Pierwszym z nich jest „oficjalna" informatyka, skupiająca się wokół wydziałów informatyki na wyższych uczelniach, placówek PAN, zakładów elektronicznej techniki obliczeniowej (ZETO), itd. Drugi nurt stanowią — upraszczając sprawę — rzemieślnicy i hobbiści.

Przedstawiciele obydwu nurtów patrzą na siebie ze skrywaną (albo i nie) niechęcią. Hobbiści spoglądają na profesjonalną informatykę pod kątem rutynowej „młócki" zbiurokratyzowanych i ociężałych ośrodków obliczeniowych oraz przestarzałego w dużej części sprzętu, pochłaniającego tony kart perforowanych i zadrukowującego koślawymi szeregami cyfr listy płac i kwity opłat energetycznych. Zawodowi informatycy widzą w hobbistach głównie niegodnych profanów, pozbawionych rzetelnych podstaw wiedzy informatycznej, lecz uzurpujących sobie prawo zabierania głosu na fachowe i niejednokrotnie trudne tematy.

Profesjonaliści mają dużo racji, zarzucając popularnym publikacjom jednostronne, nie zawsze kompleksowe i wyczerpujące przedstawianie problematyki informatycznej. Artykuły dotyczą głównie komputerów osobistych. To prawda, że komputery osobiste stanowią tylko pewien wycinek komputerowej problematyki, dlatego nie można ich fetyszyzować i uważać za panaceum na wszystkie trapiące nas informatyczne bolączki. Jest faktem, że sieć mikrokomputerów osobistych na dłuższą metę ma sens wtedy, gdy ma ona zaplecze w postaci wielkich banków danych i potężnych mocy obliczeniowych dużych komputerów w centralnych ośrodkach informatycznych. Trudno zaprzeczyć, że niektóre czasopisma i cykliczne programy telewizyjne, chcąc ugruntować swą popularność, nie oparły się koniunkturze, nadmiernie eksponując popularny aspekt informatyki, tzn. gry komputerowe.

Profesjonaliści żądają rzetelnej, skutecznej i powszechnej oświaty informatycznej społeczeństwa. Przy okazji konferencji i sympozjów pod adresem popularnych pism informatycznych rozlegają się gniewne pomruki. Zawodowi informatycy, zwłaszcza ci z akademickim cenzusem, skarżą się na lekceważenie ich opinii i brak należytej atencji ze strony wielkonakładowych czasopism i innych publikatorów.

Co przeszkadza profesjonalistom, by miast narzekać, przejąć ster oświaty komputerowej we własne ręce? W zarzucie niedopuszczania do głosu jest sporo demagogii. Każde wielkonakładowe czasopismo kupi bowiem na pniu dobry, popularnotechniczny artykuł spłodzony przez zawodowca o znanym i szanowanym nazwisku. Warunek: artykuł musi trafiać w zapotrzebowanie czytelników, być aktualny i interesująco napisany. Często pożądane są też atrakcyjne ilustracje. Sama wiedza fachowa nie stanowi jednak recepty na niezły artykuł popularny. Trzeba czasem „wyjść z siebie i stanąć obok", popatrzeć na swój profesjonalizm z przymrużeniem oka. Nie wystarczy zalać papier powodzią faktów. Konieczna jest selekcja materiałów, wyszukanie sugestywnych analogii i nieraz znalezienie rozsądnego kompromisu między naukową ścisłością, a uproszczeniami niezbędnymi dla ułatwienia percepcji. Jednym słowem, trzeba porządnie się namęczyć. Pisanie artykułów popularnych często jest trudniejsze niż specjalistycznych doniesień i rozpraw do czasopism fachowych, rozchodzących się w . nakładzie ... set egzemplarzy i trafiających do zainteresowanych i bibliotek. Abstrahując od wyższych wymagań natury językowej, trudność polega właśnie na chwilowym zerwaniu z „profesjonalnym" sposobem myślenia i „zniżeniu się" do poziomu zwyczajnego, często młodego czytelnika.

Naburmuszeni profesjonaliści zapominają, że sami ponoszą odpowiedzialność za specyficzny stosunek swego środowiska, zwłaszcza naukowego, do twórczości popularnonaukowej. Twórczość tego typu jest traktowana często z ostentacyjną pogardą, jako coś niegodnego prawdziwego naukowca. Autor popularnego artykułu słyszy często ze strony kolegów-adiunktów drwiące docinki: „ależ ty to, stary, spłyciłeś!". Publikacje popularnonaukowe nie są też w żaden sposób wliczane do dorobku pracownika nauki. Nic dziwnego, że w tej sytuacji niewielu ma ochotę „brukać swe ręce" pisaniem do pism popularnonaukowych. A jeśli już czasem coś napiszą, to często nudnawą „blachę" na niezbyt interesujące tematy. Potem narzekają, że redakcja grymasi i opiera się umieszczeniu „profesjonalnego" materiału na czołowej kolumnie. Pisanie jest rzemiosłem i — jak każde rzemiosło — wymaga praktyki. Czasem trzeba trochę poterminować. Poza tym niezbędny jest kontakt z czytelnikami, znajomość ich zainteresowań, oczekiwań, sposobu myślenia i poziomu przygotowania, tego zaś często brakuje. Znacznie wygodniej, przyjemniej i bezpieczniej jest gromić profanów z wysokości akademickiej katedry niż samemu zainwestować czas i wysiłek, opracowując nie tylko merytorycznie poprawne, ale i przystępnie napisane książki i artykuły.

Status profesjonalisty nie zapewnia dziś monopolu na „rząd dusz". Dostęp do mównicy trzeba sobie wypracować własną aktywnością. Zawodowcy mają jednak w ogólności dużo mocniejsze atuty, niż amatorzy i hobbiści. Muszą tylko chcieć.

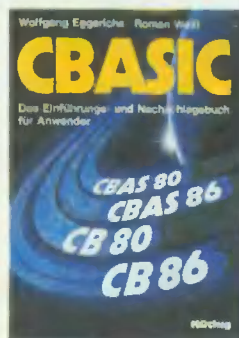
Roland Wacławek

CIEKAWE KSIĄŻKI

Guenter Juergensmeier, SO PROGRAMMIERT MAN 16-BIT-COMPUTER, Markt Technik Verlag w Haar, RFN, 1985 r.

Tytuł brzmi: Tak programuje się komputer 16-bitowy. Ta licząca 300 stron pozycja jest poświęcona programowaniu IBM-PC/XT i komputerów kompatybilnych. Książka skierowana jest do tych użytkowników IBM-PC, którzy opanowali już podstawy programowania i pragną doskonalić swe umiejętności warsztatowe. Autor wychodzi ze słusznego skądinąd założenia, że właściwego stylu i metod programowania najlepiej uczyć na dobrych przykładach. Dlatego w książce nie ma praktycznie żadnego balastu teoretycznego. Już od pierwszych stronic pojawiają się programy: od prostej gry w BASIC-u, do złożonych handlerów systemowych w języku asemblera. Książka nie jest jednak bynajmniej zbiorem listingów: każdy program jest starannie objaśniony i skomentowany, tak by Czytelnik mógł go łatwo analizować i wniknąć w jego najdrobniejsze szczegóły.

Książka zawiera m.in.: grę Reversi, różne programy usługowe w BASIC-u, usprawnienia i programy pomocnicze do współpracy z pakietami Lotus 1-2-3 i Symphony, programy asemblerowe współpracujące z językiem BASIC (m.in. wywołania systemowe z BASIC-a), filtry (specyficzny rodzaj programów usługowych, zapożyczony z systemu UNIX), ulepszony program do kopiowania ekranu na drukarkę oraz handlery monitora.



Aby zaoszczędzić Czytelnikowi zbędnego wysiłku „wklepywania" programów, do książki dołączono dyskietkę zawierającą listingi wszystkich zamieszczonych w książce programów. Pomysł kapitalny — dzięki niemu autor osiągnął swój cel, zachęcając Czytelnika do samodzielnej eksploracji komputera. Można przecież zawsze wyjść od działającej i skomento-

c.d. na str. 27

młody TECHNIK
InforMik

MAGAZYN KOMPUTEROWY
„MŁODEGO TECHNIKA"
NR. II ROCZNIK I

SPIS TREŚCI

Artykuły: MUZYKI SKOMPUTERYZOWANEJ CZĘŚĆ I — Jacek Jędrzejowski . . . 2
JAK IBM-PC OBSŁUGUJE KLAWIATURĘ? — Roland Wacławek 8
JAK ZMIEŚCIĆ JESZCZE WIĘCEJ? (gz) . . . 14
POMIĘDZY SINCLAIREM A COMMODORE — (gz) 18
ZEGAR CYFROWY W ZX SPECTRUM — Roland Wacławek 21

Felietony: NABURMUSZENI I GNIEWNI — Roland Wacławek . . . . . . II str. okl.
POD GÓRKĘ — Jerzy Klawiński 1

Działy: NASZ TEST: TIMEX PRINTER 2040 — Tadeusz Rzepecki 24
NASZ TEST: URZĄDZENIE KASETOWE DATA RECORDER 433 — Grzegorz Zalot 25
SEMINARIUM „INFORMIKA": ASEMBLER GENS 3, CZĘŚĆ II — Tadeusz Basista 28
KOMPUTER W SZKOLE: POCHODNE FUNKCJI INACZEJ — (mg) . . . 30
CIEKAWE KSIĄŻKI II str. okł.

Okładka — str. III i IV — DYSKIETKA
Numer ilustrował: Jerzy Flisak
Fotografie w numerze: Grzegorz Zalot, Władysław P. Jabłoński, Tadeusz Rzepecki, z archiwum redakcji.

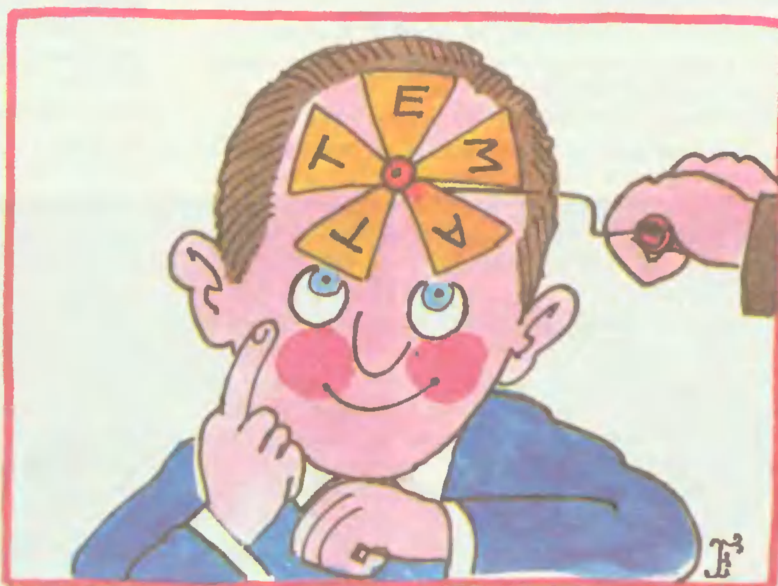
Pod GÓRKĘ

Ci, którzy uważali na lekcjach fizyki, wiedzą, że rozruch urządzenia pochłania stosunkowo najwięcej energii. Przezwyciężenie oporów podczas rozruchu jest więc zjawiskiem normalnym i należy się z nim liczyć. Najgorzej, że niektórzy ludzie z tzw. inteligencji technicznej dość gładko zapominają o tej regule w życiu i to w stosunku do wielu nowo tworzących się zjawisk.

Jednym z nich jest właśnie amatorska informatyka. I tu właśnie ciska się gromy (specjalistyczne!) na głowy popularyzatorów, że kaleczą, trywializują, spłaszczają i w ogóle — fuj! Przypomina mi to trochę amerykański szpital psychiatryczny widziany na jakimś filmie, gdzie wszyscy pielęgniarze chodzą z wysokotonowym gwizdkiem w zębach, aby w razie agresywnej postawy pacjenta jednym gwizdnięciem uruchomić system alarmowy. Otóż niektórzy spece profesjonalnokomputerowi przypominają świeżo przyjętych do pracy pielęgniarzy, którzy są tak zszokowani szpitalem i nową dla siebie sytuacją, że gwiżdżą, gdy tylko zobaczą jakiegoś obłąkańca, wprawiając całą machinę w ruch bez realnej potrzeby. Coraz to otrzymujemy pełne niepokoju listy, w których poważni spece żądają od nas wykładów na poziomie co najmniej uniwersyteckim, sugerując, że w przeciwnym wypadku czarno widzą przyszłość naszej redakcji i wychowanych na jej publikacjach młodych ludzi.

Być może nie wszystko jest jeszcze tak, jak byśmy sobie tego życzyli, ale trzeba zrozumieć, że to właśnie rozruch nowego tematu w świadomości młodzieży, jak i redaktorów „MT", pochłania znaczną część naszej energii, a zresztą proste przedstawienie rzeczy trudnych zawsze wymaga znacznego nakładu sił. Spory o metodę uważamy za bezcelowe. Być może lepiej byłoby nic nie robić, bo wtedy przynajmniej niczego się nie zepsuje, ale my — pomimo przestróg specjalistów — spróbujemy: tracąc energię pojedziemy „pod górkę" z wiarą w osiągnięcie kiedyś szczytu i w przyjemniejszą (bo z górki!) część naszej podróży, z myślą o której sprawdzamy już dziś nasze redakcyjne hamulce...

JERZY KLAWIŃSKI





MUZYKI SKOMPUTERYZOWANEJ

CZĘŚĆ 1, CZYLI ZACZNIJMY OD C64

Jacek Jędrzejowski

Muzyka była pierwszą dziedziną sztuki, do której wkroczyły komputery i w której, przynajmniej na razie, odnoszą największe sukcesy.

Początkowo wydawało się, że muzyka nie może mieć pożytku z nowych wynalazków. Bo do czegóż by miała służyć na estradzie koncertowej lampa elektronowa czy np. laser? Potem znów nastąpił gwałtowny skok elektroniki — organy elektroniczne, wzmacniacze, wieże kolumn głośnikowych. Nawet konserwatywni w swych poglądach na technikę lecz nowatorscy w odniesieniu do muzyki, kompozytorzy muzyki poważnej zaczęli najpierw nieśmiało, potem już coraz chętniej sięgać po generatory, miksery, magnetofony.

Ale „bomba" miała dopiero wybuchnąć! Gdy w roku 1946 skonstruowano pierwszy komputer elektroniczny (ENIAC), nikt jeszcze nie spodziewał się, jak wynalazek ten wpłynie na dalsze losy ludzi, a cóż dopiero na muzykę. O wiele bardziej znamienny stał się początkowo dla muzyki wynalazek (czy raczej konstrukcja) Roberta A. Mooga — syntezator. Było to w roku 1958. Już wówczas wykorzystywano komputery do tworzenia muzyki stochastycznej, czyli takiej, która co prawda podlega kształtowaniu zgodnie z działaniem przypadku, lecz kształtowanie to zdąża ku określonemu, nadrzędnemu celowi (gr. *stochos* — cel). Instrumenty zastąpiono sprzętem elektronicznym, kości do gry — komputerem. Jak wiadomo, logika maszyny cyfrowej i jej wewnętrzna organizacja jest sprecyzowana do najmniejszego detalu. Zastosowanie komputera wymaga zatem dokładnego określenia czynności, jakie ma on wykonywać. Początkowo więc, z braku odpowiednich języków programowania, starano się sformalizować muzykę. Naprzeciw tym potrzebom wyszły serializm i punktualizm (nowe kierunki w muzyce), a także aleatoryzm, który był reakcją na sztywne reguły serializmu. Od 1960 roku ruszyła seryjna produkcja syntezatorów MOOG, które szybko zaczęły opanowywać rynek muzyki rozrywkowej, znalazły się też w studiach eksperymentalnych.

W latach siedemdziesiątych daleko posunięte zostały prace nad metodami opisu zjawisk muzycznych, a co za tym idzie, specjalizowanymi językami programowania oraz nad komponowaniem muzyki stochastycznej (Uniwersytet w Illinois). Z firmą IBM współpracował znany kompozytor Iannis Xenakis, który został profesorem „muzyki matematycznej i automatycznej" w Uniwersytecie Indiana. W tym miejscu nie sposób pominąć innych wielkich

Profesjonalny, muzyczny system komputerowy PPG — połączenie syntezy analogowej i cyfrowej

kompozytorów związanych z muzyką elektroniczną, jak Pierre Boulez czy Karlheinz Stockhausen.

Wreszcie nadeszły lata osiemdziesiąte. Komputer, który dotychczas wykorzystywano do inwentaryzowania dzieł mistrzów różnych epok, do badań nowych i starych konstrukcji instrumentów, analizy fug J. S. Bacha czy mazurków Chopina, ba, któremu nawet powierzono rolę twórcy, wywołując tym w różnych kręgach zgrozę — stał się integralną częścią muzycznego instrumentarium. Powstały nowoczesne komputerowe systemy muzyczne (CMI Fairlight, Synclavier, EMU, McLevier, PPG) zdolne teoretycznie zastąpić wielką orkiestrę, a na które pozwolić może sobie obecnie prawie każde większe studio nagraniowe. W syntezatorach znalazły się mikroprocesory, umożliwiając zapamiętywanie dużych sekwencji muzycznych oraz udoskonalając sterowanie. Powstały ściśle związane z komputerami, całkiem nowe metody syntezy dźwięku. W 1982 roku narodziło się MIDI — międzynarodowy standard dotyczący sterowania cyfrowego elektronicznych instrumentów muzycznych i ich wzajemnej kompatybilności. Te wszystkie nowości przerosły wielu z tych wykonawców i realizatorów muzyki związanej ze sztuczną syntezą dźwięku, którzy nauczyli się obsługiwać stare, analogowe syntezatory. Ale tu znów przyszła z pomocą technika, oferując takie oprogramowanie nowych maszyn, które pozwala na wykorzystanie tych wszystkich nowości bez znajomości szczegółów. Nowa technika dała jeszcze coś więcej — coś, czego nie było nigdy dotychczas. Oddała w ręce szerokich mas zaawansowane narzędzia mogące służyć do tworzenia nowoczesnej sztuki, dotychczas dostępne jedynie wybrańcom losu.

Tym to wstępem chciałbym zapoczątkować cykl artykułów dotyczących najnowszych metod i urządzeń, które służą jako narzędzia w tworzeniu i realizacji muzyki. Stąd w tytule pojawia się zwrot **muzyka skomputeryzowana**, a nie **komputerowa**, gdyż ten ostatni termin sugerowałby raczej wykorzystanie komputera jako podmiotu procesu twórczego. Na razie zajmiemy się tematem niezwykle obszernym, w którym podjęta zostanie próba zarysu zagadnień począwszy od najprostszych (komputery domowe, mini-instrumenty, podstawy syntezy dźwięku) aż po bardziej złożone (profesjonalne komputerowe systemy muzyczne, instrumenty zintegrowane z komputerem, MIDI, cyfrowe metody syntezy dźwięku).

Zacznijmy więc od C64

Kontynuując niejako cykl artykułów poświęconych komputerowi Commodore 64, jakie ukazały się w „MT" chciałbym udowodnić, że ten historyczny już niemal sprzęt może służyć do bardzo ciekawych eksperymentów w zakresie muzyki. Jego główną zaletą jest to, że staje się coraz tańszy, a co za tym idzie — coraz szerzej dostępny w naszym kraju. Nie jest tajemnicą, że do jego popularności przyczynił się układ scalony do syntezy dźwięku — SID, znajdujący się wewnątrz. Nie wszyscy jednak wiedzą, jakie możliwości kryje w sobie ten układ. Gry komputerowe prezentują niejednokrotnie wysoki kunszt programistów także w zakresie muzyki, trudno jednak przypuszczać, by mogły na dłuższą metę stanowić źródło przeżyć estetycznych. Istnieje wiele programów do układania muzyki na różne sposoby: pisząc na ekranie nuty lub kody zastępcze, grając na klawiaturze komputerowej lub na muzycznej podłączonej przez interfejs. Wszystkie, z większym lub mniejszym powodzeniem, prowadzą do upatrzonego celu, tzn. do wydobycia z komputera dźwięku. Nie należy jednak przeceniać SID-a w roli poważnego narzędzia artystycznego. Wszystkie te, często bardzo drogie, zabawki pozostaną tylko namiastką prawdziwych instrumentów (nie tylko elektronicznych zresztą). Niemniej jednak SID może spełnić kilka ważnych ról prócz tej, która zadecydowała o jego obecności w C64 (dźwięk dla gier). Przede wszystkim jest doskonałym przykładem prostego syntezatora dźwięku (ang. *synthesizer*), co więcej — sterowanego cyfrowo. Ta ostatnia cecha umożliwia bardzo ciekawe eksperymenty w zakresie efektów dźwiękowych (mogących wspomagać nawet przedsięwzięcia estradowe) oraz próby uzyskiwania muzyki probabilistycznej czy wręcz stochastycznej, do badań fizjologicznych właściwości słuchu.

SID

Układ SID 6581 (*Sound Interface Device*) jest trójkanałowym (trzygłosowym) syntezatorem dźwięku zgodnym z rodziną układów mikroprocesora 6502, a także innymi mikroprocesorami. Oto jego cechy:

— 3 generatory dźwięku, od 0 do 4 kHz (8 oktaw)
— 4 rodzaje przebiegów wyjściowych w każdym generatorze
— 3 modulatory amplitudy, każdy 48 dB
— 3 generatory obwiedni (*envelope*) o parametrach:

Trzy dyskietki (spośród pięciu) zawierające zintegrowany program muzyczny MusiCalc, którego można używać np. także dla Commodore 64







— czas narastania (*attack*) od 2 ms do 8 sekund
— czas ustalania (*decay*) od 6 ms do 24 sekund
— poziom ustalenia (*sustain*) od 0 do maksymalnej głośności
— czas wygasania (*release*) od 6 ms do 24 sekund
— synchronizacja generatorów
— modulacja kołowa
— programowane filtry o parametrach:
— częstotliwość rezonansowa lub graniczna od 30 Hz do 12 kHz
— stromość zbocza 12 dB/oktawę (skuteczność)
— filtr dolno-, górno- i środkowoprzepustowy oraz środkowozaporowy
— regulacja głośności (wspólna)
— 2 przetworniki analogowo-cyfrowe (wejścia z dwóch pokręteł — *paddle*)
— wejście „Audio" (zewnętrzne źródło dźwięku)

SID posiada 29 rejestrów sterujących jego pracą. Rejestry 0—6, 7—13, 14—20 pełnią te same funkcje odpowiednio w poszczególnych trzech kanałach. Rejestry 21—24 sterują funkcjami wspólnymi dla kanałów. Rejestry 25 i 26 stanowią wyjścia przetworników analogowo-cyfrowych, 27 i 28 — wyjścia wewnętrznych stanów cyfrowych SID. Takie są suche dane techniczne. Co one oznaczają i jak je wykorzystać w praktyce dowiemy się niebawem.

UWAGA! Rejestry od 0 do 24 pozwalają tylko na wpisywanie wartości, tzn. nie można ich odczytać (*write-only*). Rejestry od 25 do 27 można tylko odczytać (*read-only*); wpisywanie w nie danych nie powoduje żadnych reakcji ze strony układu.

Ton, jego wysokość a częstotliwość fali

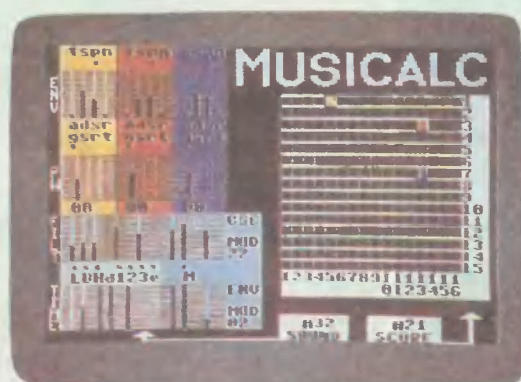
Częstotliwość fal dźwiękowych obejmuje zakres od 0 Hz do 1 GHz (tj. do miliarda drgań na sekundę), lecz słyszymy tylko te od 16 Hz do 20 000 Hz. Górna granica czułości słuchu maleje z wiekiem.

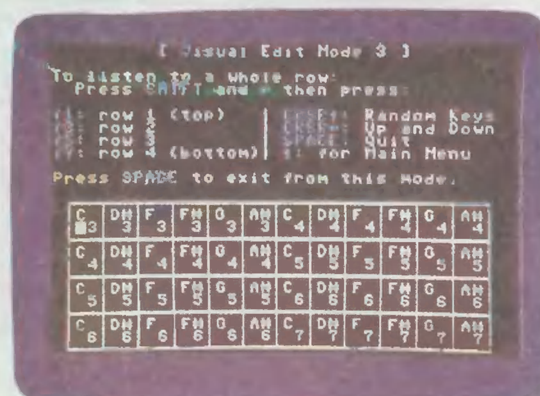
Dźwiękiem nazywamy wszystkie zjawiska dźwiękowe rejestrowane przez ludzki narząd słuchu. Jeśli jednak dźwięk jest wywołany przez drganie proste (takie, które jest efektem ruchu harmonicznego prostego, czyli będącego rzutem ruchu jednostajnego po kole) i przenoszony przez fale o przebiegu ściśle sinusoidalnym, nazywamy go **tonem**. Wysokość tonu może być jednoznacznie określona przez częstotliwość fali. Muzyka jednak nie operuje częstotliwościami, gdyż rozpiętość tychże jest bardzo duża, a wrażenia wysokości tonu nie zmieniają się proporcjonalnie do częstotliwości, lecz do jej **logarytmu** (prawo Webera-Fechnera). Obecnie używanym wzorcem stroju muzycznego jest ton o częstotliwości 440 Hz, odpowiadający dźwiękowi 'a^1' (*a razkreślne*). Teoria muzyki i pismo muzyczne określają wysokość tonów nazwami literowymi i znakami na pięciolinii. Podstawowy interwał muzyczny (odległość między dźwiękami, a ściślej — między stopniami gamy) — **oktawa**, odpowiada stosunkowi częstotliwości 2:1. Dźwięki w kolejnych oktawach posiadają te same nazwy literowe, lecz różne wysokości, tzn. dźwięki następnej oktawy wyższej mają dwa razy większe częstotliwości. Od czasów starożytnych panowały tendencje podziału oktawy na 12 równych interwałów (części). W XVIII wieku na bazie teoretycznych prac Neihardta powstał system **równomiernie temperowany dwunastodźwiękowy**. W granicach oktawy system ten posiada 12 dźwięków, odległych od siebie o jednakowy interwał — **półton temperowany**. W porównaniu do panujących uprzednio systemów nierównomiernie temperowanych system ten daje możliwość równorzędnego użycia wszystkich tonacji i akordów oraz upraszcza konstrukcje niektórych instrumentów, choć jest pewnym kompromisem. Wielkość oktawy pozostaje tu bez zmian w stosunku do systemów nietemperowanych — współczynnik wynosi zawsze 2. Pozostałe interwały (oprócz prymy) mają współczynniki niewymierne. W 1873 roku wprowadzono do akustyki nową jednostkę zwaną cent. Każdemu półtonowi odpowiada 100 ct (centów). I po co nam to wszystko? Otóż właśnie...

Algorytm definicji skali muzycznej

Każdemu posiadaczowi C64 chyba jest znana tabela znajdująca się na końcu instrukcji obsługi komputera, zawierająca 94 nazwy dźwięków w 8 oktawach, odpowiadające im parametry liczbowe (16-bitowe) oraz ich 8-bitowe postacie (młodszy i starszy bajt), wreszcie częstotliwości. Dla każdego dźwięku trzeba wpisywać w rejestry układu dwa wspomniane powyżej bajty, tzn. liczby nie mające kompletnie nic wspólnego z nazwą dźwięku czy jego zapisem literowym bądź nutowym,

Program MusiCalc na ekranie monitora komputerowego







Najmniejszy, „zabawowy" wręcz zestaw do muzyki komputerowej

prócz tego tylko, że określają wysokość danego dźwięku. Chcąc zaprogramować jakąś melodię, trzeba jej zapis sformalizować, czyli zastąpić nazwy dźwięków liczbami, co wiąże się ze żmudnymi poszukiwaniami w tablicy. Jest to metoda, w której możemy łatwo popełnić błąd. Ale można postąpić także inaczej, tzn. zmusić komputer by „rozumiał" nazwy dźwięków. W tym celu trzeba najpierw zdefiniować skalę muzyczną, tzn. skalę wartości liczbowych będących parametrami określającymi wysokości generowanych dźwięków.

Półton posiada niewymierny współczynnik liczbowy, jako że jest jedną z dwunastu równych części oktawy. Współczynnik ten jest równy $2^{1/12}$ (dwa do potęgi jedna dwunasta). Fakt ten należy wykorzystać w procedurze definiującej skalę muzyczną. Program nr I jest tu przykładem. Skorzystanie z niego będzie o wiele wygodniejsze od przepisywania wartości z tabeli. Po pierwsze daje możliwość dostrojenia skali dźwiękowej SID do własnych potrzeb, wspomniany bowiem wzorzec 440 Hz nie jest bezwzględnie obowiązujący. Poza tym w niektórych książkach i amerykańskich instrukcjach podane są wartości tabeli dla innej niż 0.98 MHz częstotliwości zegara systemowego (tu dźwiękowi C-0 odpowiada wartość parametru 228), jasne więc, że wpisując takie dane uzyskamy inny od wzorcowego strój. Po drugie, za pomocą krótkiego algorytmu definiuje całą skalę muzyczną (równomiernie temperowaną). Po trzecie, umożliwia zapis dźwięków za pomocą zgodnych z terminologią muzyczną nazw literowych.

Nim pokrótce omówimy program, najpierw pewna istotna uwaga: dźwięki w tabeli oznakowane są od C-0 do A-7, co ma oznaczać kolejne oktawy. Nie jest to zgodne z nomenklaturą muzyczną, według której np. dźwięk a-razkreślne (440 Hz) oznacza się jako 'a^1' a nie 'A-4' jak w tabeli. Odpowiednio dla dźwięków w tabeli oznaczenia muzyczne byłyby następujące (dźwięki przyjmują w nazwach takie samo zakończenie jak oktawa, w której się znajdują):

TABELA 1

oktawa subkontra:	C_2 do H_2 (w tabeli C-0 do H-0)
oktawa kontra:	C_1 do H_1 (w tabeli C-1 do H-1)
oktawa wielka:	C do H (w tabeli C-2 do H-2)
oktawa mała:	c do h (w tabeli C-3 do H-3)
oktawa razkreślna:	c^1 do h^1 (w tabeli C-4 do H-4)
oktawa 2-kreślna:	c^2 do h^2 (w tabeli C-5 do H-5)
oktawa 3-kreślna:	c^3 do h^3 (w tabeli C-6 do H-6)
oktawa 4-kreślna:	c^4 do h^4 (w tabeli C-7 do A-7)

(w muzyce używa się jeszcze oktaw 5- i 6-kreślnej).

Dźwięki, po których następuje znak # (np. C-3) oznaczają dźwięki podwyższone o półton i według terminologii muzycznej otrzymują końcówkę 'is' (cis małe). Podobnie, dźwięki obniżone kończą się na 'es' (za wyjątkiem H, który po obniżeniu nazywa się B). W programowaniu wygodniej posługiwać się jednak notacją „komputerową", licząc oktawy od zera.

Przejdźmy zatem do programu. Fragment inicjujący pracę SID na razie pominiemy, gdyż zagadnienia te zosta-

ną omówione potem bardziej szczegółowo. Fragment o numerach linii 190 do 250 dotyczy uzyskania dźwięku wzorcowego, którego początkowa częstotliwość określona jest na 440 Hz (a^1), a jego parametr zgodnie z tabelą wynosi 7493 (można to sprawdzić doświadczalnie posługując się kamertonem). Działa tu pętla, zwiększająca lub zmniejszająca wartość zmiennej po naciśnięciu jednego z klawiszy funkcyjnych. Klawisze F1 i F7 służą tu do regulacji zgrubnej (w górę lub w dół), a F3 i F5 — dokładnej. Wartość parametru jest natychmiast przeliczana na częstotliwość (w Hz) i wyświetlana. Następnie, za pomocą uprzednio zadeklarowanych w linii 140 funkcji jest obliczany starszy i młodszy bajt parametru, po czym wartości te są wpisywane do odpowiednich rejestrów SID, decydujących o wysokości dźwięku. Zmienna *SI* „pamięta" adres bazowy kontrolera dźwięku. Najistotniejszy na razie jest fragment programu od linii 290 do linii 330.

W linii 290 obliczana jest wartość parametru dla najniższego dźwięku przyszłej skali. Dźwięk ten, to *C subkontra* (C-0). Jest on oddalony od *a-razkreślnego* o 4 oktawy i 9 półtonów w dół. Stąd wyrażenie (4+9/12), które jest wykładnikiem liczby 2, dając w ten sposób współczynnik liczbowy odległości. Odległość ta liczona jest w relacji do dźwięku 'a^1', którego parametr znajduje się w zmiennej *PA*. Zatem w zmiennej *P0* otrzymujemy wartość całkowitą parametru dźwięku 'C_2'. Jeśli 'a^1' ma 440 Hz, to 'C_2' ma 16.35 Hz (parametr = 228). Zmiennej *WP* w dalszej części tej linii przypisana jest wartość współczynnika liczbowego półtonu temperowanego, czyli 1/12 części oktawy. W linii 300 „startuje" pętla obliczająca dokładnie wartości parametrów wszystkich 95 dźwięków skali możliwych do uzyskania przy pomocy SID. Odbywa się to przez pomnożenie wartości parametru dźwięku podstawowego (C_2) przez kolejne współczynniki odległości następnych 94 dźwięków skali. Współczynniki te otrzymujemy podnosząc współczynnik półtonu do potęgi o wykładniku równym aktualnej wartości zmiennej pętli (czyli numerowi kolejnego dźwięku). Niestety, skala parametryczna SID nie jest idealnie zbieżna wykładniczo z częstotliwością generowanych dźwięków. Dlatego funkcja wykładnicza f(J) = P0 * WP J wymaga korekcji. Na podstawie dość żmudnych analiz zastosowałem korekcję liniową tej funkcji, przedstawiającą się wyrażeniem PM/(2 * P0+15) uzależniając jej wielkość od parametru początkowego PO. Wprowadzone wyrażenie dało nadspodziewanie dobry skutek — przy parametrze początkowym równym 228 (440 Hz) skala idealnie pokrywa się z tabelą ustaloną przez producenta SID. Dodanie wartości 0.5 jest konieczne ze względu na działanie funkcji INT (), która nie zaokrągla, lecz obcina część ułamkową liczby.

W linii 320 trzem zmiennym indeksowanym są przypisywane kolejno: wartości młodszych bajtów (NL(J)) i starszych bajtów (NH(J)) parametrów. Właśnie wartości dwóch ostatnich zmiennych służą do określania wysokości dźwięków skali muzycznej. W linii 360 wartości te wprowadzane są do rejestrów SID. Jeśli podłączymy komputer do wzmacniacza — usłyszymy dźwięki całej skali.

Fragment programu o numerach linii od 280 do 320 może być wykorzystany w dowolnym programie z zastosowaniem dźwięków muzycznych. Zajmie on o wiele mniej miejsca w pamięci (4 krótkie linie!), niż dane z tabeli zapisane w liniach DATA. Można tu spróbować ciekawych eksperymentów, definiując inne skale muzyczne (np. ćwierćtonową). Należy tylko zmiennej *PO* przypisać bezpośrednio (zamiast wyrażenia) jakiś parametr początkowy (np. 228) określający najniższy dźwięk skali. Można także wartości zmiennych indeksowanych NL () i NH () zapisać na dyskietce lub taśmie i ładować je bezpośrednio z pamięci zewnętrznej do swego programu. Pozostawiam te eksperymenty pomysłowości Czytelnika.

TABELA 2

Wartość		Narastanie (ATTACK)	Ustalanie i wybrzmiewanie DECAY/RELEASE
DEC	HEX	czas trwania fazy	czas trwania fazy
0	0	2 ms	6 ms
1	1	8 ms	24 ms
2	2	16 ms	48 ms
3	3	24 ms	72 ms
4	4	38 ms	114 ms
5	5	56 ms	168 ms
6	6	68 ms	204 ms
7	7	80 ms	240 ms
8	8	100 ms	300 ms
9	9	250 ms	750 ms
10	A	500 ms	1.5 s
11	B	800 ms	2.4 s
12	C	1 s	3 s
13	D	3 s	9 s
14	E	5 s	15 s
15	F	8 s	24 s

Nasze zamiary szły jednak jeszcze dalej. Otóż komputer ma rozpoznawać literowe nazwy dźwięków. Zadanie to realizuje procedura rozpoczynająca się w linii 550. Przetwarza ona dane źródłowe z linii DATA na wartości liczbowe, tzn. mniej i bardziej znaczące bajty parametrów wysokości dźwięku. Wartości te przechowywane są w tablicach (zmiennych indeksowanych TL () i TH (). Jest to proces w rodzaju kompilacji. Gotowe dane wykorzystywane są we fragmencie programu o numerach linii od 400 do 510, który zajmuje się odtwarzaniem muzyki. Pozwala to na uzyskanie nawet w BASIC-u zadowalającego tempa utworu przy niezależnym wykorzystaniu trzech głosów równolegle (!).

Oto krótkie wyjaśnienie zasad zapisu w liniach DATA. Każdy głos zapisujemy oddzielnie, uważając jednak, by wartości dźwięków (czy trwania) były prawidłowe (w przeciwnym razie głosy będą się „rozbiegały"). Dźwięki zapisujemy literowymi nazwami (dla dźwięków obniżonych stosujemy enharmoniczne zamienniki). Dodatkowo do każdego dźwięku dopisujemy oznaczenie oktawy według wspomnianego „komputerowego zapisu", kierując się ich relacjami do zapisu nutowego podanymi uprzednio w tabeli. Dane liczbowe sąsiadujące z nazwami dźwięków oznaczają wartości metryczne dźwięków (półnuta = 8, ćwierćnuta = 4, ósemka = 2, szesnastka = 1). Litera 'P' oznacza pauzę, dla której wartości metryczne są te same co dla nuty.

Omówione dane odczytywane są sekwencyjnie w linii 560. Dalsze czynności omawianej procedury to: rozpozna-

nie oktawy (zmienna *OK*) i wpisanie wartości metrycznej nuty do tablicy (linia 570), rozpoznawanie nazwy dźwięku i kontrola poprawności zapisu (linie 580—600), oraz wpisanie do tablic wartości bajtów parametru dla zidentyfikowanego dźwięku (linia 620) lub pauzy (61). Każdy głos analizowany jest osobno.

W linii 400 programu jest zdefiniowany łańcuch dopuszczalnych nazw dźwięków oraz pauzy. Zmienna *MX* określa potrzebną ilość miejsc w tablicach. W linii 420 definiowane są adresy rejestrow zmieniających rodzaj generowanego przebiegu i wartości odpowiadające różnym przebiegom (zostanie to omówione za chwilę). Linia 440 ustawia wymagane wartości rejestrów SID, zaś linia 450 trzykrotnie wywołuje omówioną powyżej procedurę kompilującą. Linie od 470 do 510 to „mechanizm grający". Oblicza on wartości metryczne nut w każdym głosie oddzielnie (linia 490), a w odpowiednim momencie wyłącza lub włącza dźwięki (linie 480 i 510). Uprzedzamy jednak pewne sprawy, które trzeba omówić bardziej dokładnie.

Czas trwania dźwięku a wartość metryczna nuty

Czas trwania różnych krótkich zjawisk mierzy się zazwyczaj w milisekundach lub sekundach. Dla celów muzycznych istotne jest określenie względnego czasu trwania poszczególnych dźwięków. Pewne niuanse czasowe wynikają z interpretacji muzycznej i te pominiemy. Teoria muzyki i pismo muzyczne operują pojęciem wartości metrycznej dźwięku, która jest wyznaczona przez tempo utworu i wartość nuty.

Wiemy już trochę o rejestrach SID. Otóż rejestry 4, 11 i 18 (dla kanałów 1, 2, 3) odpowiednio decydują o kontroli generatorów. Każdy bit włącza tu inną funkcję (w nawiasach podane będą wartości dziesiętne odpowiadające stanowi wysokiemu danego bitu). Na razie istotny jest dla nas bit 0, czyli najmłodszy. Jest to bit „bramkujący" sygnał wyjściowy generatora (jest jakby przełącznikiem). Jeśli stan jego jest niski (0), to na wyjściu danego kanału brak jest sygnału, jeśli wysoki (1) — sygnał jest. Stanem tego bitu wpływamy na czas trwania dźwięku. Prócz tego, by pojawił się sygnał na wyjściu generatora, należy ustalić rodzaj generowanego przebiegu i obwiednię dźwięku. O rodzaju przebiegu decydują bity 4, 5, 6 i 7, czyli cztery najstarsze. Odpowiada to kolejno wartościom dziesiętnym 16, 32, 64 i 128 (2^4, 2^5, 2^6, 2^7). Chcąc uzyskać sygnał o określonym „kształcie" zerujemy bit bramkujący, a następnie ustawiamy go wraz z bitem określającym rodzaj przebiegu. Aby to uczynić, należy zsumować tzw. wagi dziesiętne odpowiednich bitów i wartość tę wpisać do rejestru. Na przykład, jeśli chcemy w kanale pierwszym uzyskać przebieg trójkątny (bit 4), wpisujemy w rejestr 4 wartość 17 (16+1, patrz linia 160 programu). Wpisanie teraz wartości 0 w ten rejestr spowoduje wyłączenie generatora. Jeśli więc chcemy uzyskać sekwencje dźwięków następujących po sobie (muzyka), to zerowanie tego rejestru spowoduje za każdym razem słyszalny trzask. By tego uniknąć, trzeba zerować tylko bit bramkujący. W wypadku przebiegu trójkątnego odpowiada to wpisywaniu na przemian wartości 16 (16+0) i 17 (16+1). Generator pozostaje wówczas cały czas włączony, a tylko sygnał będzie blokowany (rozkaz POKE w linii 480). Podsumowując można stwierdzić, że początek i koniec trwania nuty wyznacza impuls bramkujący, rozpoczynający się w momencie ustawienia bitu bramkującego i kończący się w momencie jego wyzerowania. Nie jest to jednoznaczne z zakończeniem trwania dźwięku. O tym jednak, jak i o wielu innych elementach syntezy dźwięku oraz o pozostałych własnościach układu SID dowiedzą się Czytelnicy z drugiej i trzeciej części tego artykułu.

```
100 REMINISCENCJE MUZYCZNE W JEZYKU BASIC * (C) JACEK JEDRZEJOWSKI 1986
110 :
120 REM ------ DEFINICJE ZMIENNYCH I FUNKCJI , ZAINICJOWANIE SID ------
130 DIM  P(95),NL(95),NH(95) :SI=54272 :Q(0)=-10:Q(1)=-1:Q(2)=1:Q(3)=10
140 DEF FN H(F)=INT(F/256) : DEF FN L(F)=F- FN H(F) *256
150 POKE SI+4,0: POKE SI+11,0: POKE SI+18,0: POKE SI+5,51:POKE SI+6,243
160 POKE SI+24,10: POKE SI+4,17: POKE 650,255
170 :
180 REM"----------- STROJENIE DZWIEKU WZORCOWEGO DLA SKALI ------------
190 PA=7493 : CZ=440 : REM"PARAMETR I CZĘSTOTLIWOSC - WART. POCZATKOWE
200 PRINT CHR$(147);"UZYWAJAC KLAWISZY FUNKCYJNYCH DOSTROJ DZWIĘK A ."
210 PRINT CHR$(19);CHR$(17);CHR$(17);"A-4 =",CZ,CHR$(157);" HZ       "
220 CZ=PA*0.0587214734
230 POKE SI,FN L(PA) : POKE SI+1,FN H(PA)
240 GET G$ : G=ASC(G$+CHR$(0)) : IF G>132 AND G<137 THEN PA=PA+Q(138-G)
250 IF G<>13 THEN 210
260 POKE SI+4,0 : PRINT"DEFINIOWANIE SKALI MUZYCZNEJ - PROSZĘ CZEKAC!"
270 REM ************************************************************
280 REM"--- DEFINICJA SKALI MUZYCZNEJ W PEŁNYM ZAKRESIE TONOW (!!!) ---
290 P0=INT(PA/2↑(4+9/12)) : WP=2↑(1/12)
300 FOR J = 0 TO 94
310 PM=INT(P0*WP↑J+0.5+PM/(2*P0+15)) : REM" (C)JJ /PRAWA ZASTRZEZONE./
320 P(J)=PM: NL(J)=FN L(PM): NH(J)=FN H(PM)
330 NEXT J
340 REM ************************************************************
350 REM --- DEMO ---
360 FOR J=0 TO 94: POKE SI,NL(J):POKE SI+1,NH(J):PRINT P(J),NL(J),NH(J)
370 POKE SI+4,17:FOR T=0 TO 100:NEXT T: POKE SI+4,16:NEXT J:POKE SI+4,0
380 :
390 REM"----- ROZPOZNAWANIE DZWIEKOW NA PODSTAWIE NAZW LITEROWYCH -----
400 N$="C C#D D#E F F#G G#A A#H P ": REM"ZNAKI 12 DZWIEKOW GAMY I PAUZY
410 MX=110 : DIM  TL(2,MX), TH(2,MX), DP(2,MX)
420 FOR G=0 TO 2 : W(G)=SI+G*7+4 : NEXT G : V(0)=17 : V(1)=65 : V(2)=33
430 :
440 FOR J=0 TO 24 : READ D : POKE SI+J,D : NEXT J : REM  USTAWIENIE SID
450 FOR G=0 TO 2 : GOSUB 550 : NEXT G : REM"PRZETWORZENIE TRZECH GŁOSOW
460 FOR G=0 TO 2 : L(G)=-1 : M(G)=0 : NEXT G
470 FOR G=2 TO 0 STEP-1 : REM"PETLA GŁOWNA ODTWARZAJACA UTWOR MUZYCZNY -
480 IF M(G)=0 THEN L(G)=L(G)+1: M(G)=DP(G,L(G)): POKE W(G),V(G) AND 254
490 M(G)=M(G)-1 : IF M(G)=-1 THEN EN=EN+1 : IF EN=3 THEN EN=0: GOTO 460
500 L=L(G) : POKE W(G)-4,TL(G,L) : POKE W(G)-3,TH(G,L) : NEXT G
510 POKE W(0),V(0): POKE W(1),V(1): POKE W(2),V(2): GOTO 470: REM -----
520 END
530 :
540 REM"- PROCEDURA PRZETWARZAJACA DANE POJEDYNCZEGO GŁOSU UTWORU MUZ.-
550 L=0 : PRINT"PRZETWARZANIE DANYCH DLA GŁOSU ";G+1
560 READ D$,D : IF D$="*" THEN RETURN
570 OK=VAL(RIGHT$(D$,1)) : DP(G,L)=D : J=-1
580 D$=LEFT$(D$,2): IF RIGHT$(D$,1) <> "#" THEN D$=LEFT$(D$,1)+CHR$(32)
590 IF J>12 THEN PRINT"BŁAD ZAPISU W GŁOSIE:";G+1;" POZYCJA:";L+1 :STOP
600 J=J+1 : IF D$ <> MID$(N$,J*2+1,2) THEN 590
610 IF J=12 THEN TL(G,L)=0 : TH(G,L)=0 : L=L+1 : GOTO 560 : REM PAUZA
620 NR=OK*12+J : TL(G,L)=NL(NR) : TH(G,L)=NH(NR) : L=L+1 : GOTO 560
630 :
640 REM"--- DANE DLA REJESTROW SID ---
650 DATA 0,0,0,0,0,18,146,0,0,203,0,0,26,0,0,0,0,0,0,106,32,0,50,244,47
660 :
670 REM"--- MUZYKA - GŁOS 1 ---
680 DATA A4,4,D5,6, C#5,1,D5,1,E5,2,G4,2, G4,2,F#4,2,P,2,D5,2
690 DATA F#4,2,E4,2,P,2,C#5,2, E4,2,D4,2,P,2,H4,2, D4,2,C#4,2,P,2,A4,2
700 DATA H3,2,C#4,2,D4,5, C#4,1,D4,1,E4,1,D4,1,C#4,1,D4,1
710 DATA E4,1,D4,1,C#4,1,H3,1,A3,1,G3,1,F#3,1,E3,1
720 DATA F#3,2,F#4,1,G4,1,A4,1,G4,1,F#4,1,G4,1
730 DATA A4,1,G4,1,F#4,1,E4,1,D4,1,C4,1,H3,1,A3,1
740 DATA H3,1,D4,1,E4,1,F#4,1,G4,1,H4,1,C#5,1,D5,1
750 DATA E5,1,G4,1,F#4,1,E4,1,E5,1,D5,1,C#5,1
760 DATA D5,1,F#4,1,G#4,1,A4,1,G#4,1,D5,1,C#5,1,H4,1
770 DATA C#5,1,A4,1,H4,1,C#5,1,D4,1,C#5,1,H4,1,A4,1
780 DATA G#4,1,E4,1,F#4,1,G#4,1,A4,1,H4,1,C#5,1,D5,1, E5,4,A5,6
790 DATA G#5,1,A5,1,H5,2,D5,2,C#5,2,E5,2,D5,1,C#5,1,H4,1,A4,1,A4,10,*,0
800 :
810 REM"--- MUZYKA - GŁOS 2 ---
820 DATA F#4,8, E4,12, D4,8, C#4,8, H3,8, A3,8, G#3,4, A3,16, P,8, P,8
830 DATA P,8,P,8,P,8,P,8,P,8, C#5,8, D4,10, A4,4,G#4,2, E4,1,C#4,8, *,0
840 :
850 REM"--- MUZYKA - GŁOS 3 ---
860 DATA D3,2,E3,2,F#3,2,G3,2, A3,2,H3,2,C#4,2,A3,2
870 DATA D4,2,C#4,2,H3,2,A3,2, G3,2,E3,2,A3,2,G3,2
880 DATA F#3,2,H3,2,G#3,2,E3,2, A3,2,G#3,2,F#3,2,E3,2
890 DATA D3,2,C#3,2,H2,2,E3,2, A2,2,H2,2,C#3,2,H2,2
900 DATA A2,2,H2,2,C#3,2,A2,2, D3,2,E3,2,F#3,2,E3,2
910 DATA D3,2,E3,2,F#3,2,D3,2, G3,2,F#3,2,E3,2,D3,2
920 DATA C#3,2,H2,2,A#2,2,F#2,2, H2,2,E2,2,E3,2,G#3,2
930 DATA A3,2,G#3,2,F#3,2,D3,2, E3,2,D3,2,C#3,2,H2,2
940 DATA A2,2,H2,2,C#3,2,D3,2, E3,2,F#3,2,G#3,2,E3,2
950 DATA A3,2,C#3,2,D3,2,E3,2, A2,3,E3,1,A3,6, *,0
READY.
```



MONITOR Z PŁASKIM EKRANEM DLA KOMPUTERA OSOBISTEGO

Najbardziej konserwatywnym technologicznie elementem większości nowoczesnego sprzętu informatycznego, jest bez wątpienia monitor ekranowy. Podczas gdy generacje elementów mikroelektronicznych: mikroprocesorów, pamięci itd., zmieniają się co parę lat, we wnętrzu przestronnej obudowy monitora nieodmiennie rozpiera się baniasty kineskop. Prawda, że pod względem kontrastu, małej bezwładności, prostoty sterowania i możliwości barwnej prezentacji ciągle jeszcze przewyższa on coraz liczniejszą konkurencję w postaci wyświetlaczy plazmowych, elektroluminescencyjnych i ciekłokrystalicznych (LCD). Duże wymiary, ciężar i energochłonność kineskopu stanowią jednak dokuczliwe wady, zwłaszcza w obliczu dążeń do uczynienia sprzętu informatycznego przenośnym i poręcznym.

W komputerach „podręcznych" płaskie wyświetlacze, głównie LCD, stały się już standardem. Ich ekspansja w systemach biurkowych została zahamowana gorszą czytelnością ekranu, silnie zależną od oświetlenia zewnętrznego oraz mniejszą powierzchnią roboczą niż w konwencjonalnym monitorze. Pewną przeszkodą był też całkowicie inny sposób sterowania kineskopu i wyświetlaczy mozaikowych. Ostatnie osiągnięcia w dziedzinie wyświetlaczy LCD zwiastują jednak w dającej się przewidzieć przyszłości zmierzch kineskopów — przynajmniej w monitorach monochromatycznych. Za przykład niech posłuży monitor ekranowy LCD o nazwie **Flat-Screen**, produkowany przez norweską firmę ASK. Rozmiary ekranu odpowiadają typowemu monitorowi z kineskopem o przekątnej ekranu 12 cali. Pobór mocy: zaledwie 50 mW. Niewielki ciężar monitora (tylko 1,2 kg) pozwolił zawiesić go na przegubowym wysięgniku — jak lampę kreślarską. Dzięki temu monitor praktycznie nie zajmuje cennego miejsca na biurku. Użytkownik może swobodnie przemieszczać ekran w polu widzenia i wybrać jego najdogodniejsze z punktu widzenia ergonomii położenie. Przegubowe zawieszenie zneutralizowało też jedną z głównych wad ekranów LCD, jakim jest ograniczony kąt obserwacji (w praktyce nie więcej niż 30 stopni). Ekran **Flat-Screen** można przecież ustawić zawsze prostopadle do osi widzenia.

Monitor **Flat-Screen** przeznaczony jest dla mikrokomputerów klasy IBM PC XT i AT, zapewniając rozdzielczość graficzną 640 × 200 punktów lub 25 wierszy po 80 znaków. Odpowiada to standardowej karcie graficznej CGA w trybie monochromatycznym. **Flat-Screen** ma własny pakiet graficzny, wstawiany do komputera zamiast karty CGA i zgodny z nim programowo. Z punktu widzenia oprogramowania obsługa monitora **Flat-Screen**, jest więc identyczna, jak w przypadku karty CGA, współpracującej ze zwykłym monitorem.

Dodatkową zaletą płaskich wyświetlaczy LCD i plazmowych jest eliminacja szkodliwego promieniowania kineskopu oraz męczącego dla wzroku migotania obrazu, związanego z jego periodycznym odświeżaniem. Dość wysoka cena płaskich ekranów z pewnością ulegnie obniżce w miarę postępów technologii i rozszerzenia ich produkcji. **(rw)**



JAK IBM-PC OBSŁUGUJE KLAWIATURĘ!

Roland Wacławek

Jedną z najbardziej charakterystycznych cech koncepcji mikrokomputera IBM-PC (pod tą nazwą będziemy rozumieli także kopie lub prawie kopie rodem z Tajwanu itd.), jest sposób dołączenia klawiatury (ang. *keyboard*).

We wszystkich praktycznie komputerach domowych i w większości wcześniejszych, profesjonalnych komputerow osobistych (np. Apple), klawiaturę obsługiwał główny mikroprocesor. Oprócz pewnego zmniejszenia tempa pracy całości systemu powodowało to i tę niedogodność, że klawiatura musiała być praktycznie zintegrowana z mikrokomputerem. Jeśli mikroprocesor musiał sam testować stan poszczegolnych klawiszy (ściślej: grup klawiszy), to klawiatura musiała być połączona z mikrokomputerem wiązką kilkunastu przewodow, które nie mogły byc zbyt długie z uwagi na opóźnienia i niebezpieczeństwo zakłocen.

Mikrokomputer IBM-PC został wyposażony w klawiaturę autonomiczną, wyposażoną w własny mikrokomputer, realizujący wszystkie zadania związane z testowaniem stanu przyciskow, automatycznym powtarzaniem emisji znaku w przypadku dłuższego wciśnięcia (autorepeat), itd. „Inteligentna" klawiatura umożliwia też skuteczne wyeliminowanie wszelkich efektow związanych z wibracją zestyków klawisza, równoczesnym wciśnięciem kilku przyciskow, przyciśnięciem kolejnego klawisza przed zwolnieniem poprzedniego, itd.

Klawiatura połączona jest z komputerem długim, elastycznym, pięciożyłowym kablem (normalnie wykorzystywane są tylko cztery żyły). Kabel ten zarowno dostarcza z komputera napięcia zasilającego +5V, jak rownież służy do transmisji danych od klawiatury do komputera (w pewnych warunkach — także i w przeciwną stronę). Kabel jest zakończony znormalizowanym, pięciobolcowym gniazdem DIN (diodowym), dzięki czemu dowolną klawiaturę można połączyć z dowolnym komputerem — pod warunkiem, że oba elementy należą do rodziny IBM-PC.

Klawiatura do IBM-AT funkcjonuje podobnie. Z uwagi na drobne różnice w systemie transmisji nie jest jednak możliwa poprawna eksploatacja klawiatury od PC XT z komputerem AT i odwrotnie. Na rynku spotykane są jednak coraz częściej klawiatury dostosowane do pracy w obydwu standardach. Przełączanie trybu pracy odbywa się przełącznikiem, dostępnym przeważnie od spodu klawiatury. Standardowa klawiatura do PC XT liczy 83 klawisze, do AT — 84. Spotykane są też bardziej rozbudowane i ergonomiczne klawiatury z oddzielnymi klawiszami sterowania kursorem, dodatkowymi, często programowalnymi klawiszami funkcyjnymi, itd. Ponieważ jednak wstępną obróbką danych zajmuje się wewnętrzny mikroprocesor klawiatury, do komputera PC XT docierają tylko przetworzone wstępnie informacje o wciśnięciu określonych klawiszy. PC XT „nie zauważa" więc zmiany wariantu klawiatury — jest to jedna z istotnych zalet zastosowanego rozwiązania.

Rys. 2 ukazuje schemat ideowy typowej klawiatury do IBM-PC/XT. Jak widać, konstrukcja mikrokomputera, opartego o mikroprocesor jednoukładowy INTEL

8048, została uproszczona do absolutnego minimum. Spotykane są zresztą i inne rozwiązania, z pokrewnymi mikroprocesorami 8049, 8749 lub np. z mikroprocesorem 8039 i zewnętrzną pamięcią ROM, często występujące zwłaszcza w przełączanych klawiaturach XT AT z uwagi na potrzebę większego obszaru pamięci ROM, niż dostępny w 8048. Klawiatura ma niewielki wewnętrzny bufor znaków, który może przechować dane o 20..30 uderzeniach w klawisz w przypadku, gdyby chwilowo komputer PC XT nie był przygotowany do ich odbioru. Dzięki temu w normalnych warunkach operator nie musi stale zwracać uwagi, czy komputer rejestruje wciśnięte klawisze. W IBM-PC informacja o naciśniętych klawiszach nie jest więc stracona nawet w czasie pisania tzw. ślepego, tzn. wtedy, gdy na ekranie nie pojawia się natychmiast „echo" wciśniętych klawiszy.

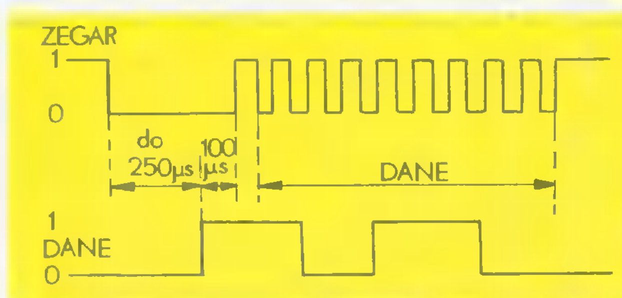
Transmisja danych odbywa się w sposób szeregowy, synchroniczny, bajt po bajcie. Na linii zegarowej w odstępach 100 mikrosekund pojawiają się impulsy zegarowe, powodujące w odbiorniku odczyt stanu linii danych (rys. 1).

Jaki charakter mają dane, przesyłane od klawiatury do komputera? Rozwiązaniem najprostszym, niemalże narzucającym się i powszechnie stosowanym, np. w klawiaturach produkcji MERA-ELZAB, jest transmisja znaków ASCII, odpowiadających naciśniętym klawiszom. Tymczasem kody przesyłane do PC/XT nie mają nic wspólnego z kodem ASCII, a nawet niekoniecznie muszą sygnalizować wprost naciśnięcie klawisza. Pozornie wygląda to na komplikację, w rzeczywistości decyduje jednak o elastyczności i walorach funkcjonalnych klawiatury PC/XT.

Wszystkie przyciski klawiatury PC/XT są ponumerowane w pewien standardowy, arbitralnie przyjęty sposób, odzwierciedlający raczej fizyczną organizację klawiatury niż alfabet lub kod ASCII. Np. klawisz **[Q]** ma numer 16, **[W]** — 17, **[E]** — 18, **[Ctrl]** — 29, lewy klawisz **[Shift]** — 42, a prawy — 54.

Klawiatura melduje komputerowi fakt wciśnięcia i zwolnienia każdego klawisza. To nie pomyłka: do komputera przesyłany jest oddzielny kod zarówno w przypadku naciśnięcia, jak i zwolnienia każdego klawisza. W rzeczywistości każdy klawisz posiada więc nie jeden, lecz dwa kody, sygnalizujące wciśnięcie i zwolnienie. W praktyce kod wciśnięcia jest po prostu numerem klawisza, a kod zwolnienia jest większy od kodu wciśnięcia o 128,

Rys. 1



co odpowiada ustawieniu w bajcie kodu najstarszego bitu. Przykładowo: wciśnięciu klawisza **[W]** odpowiada kod 17 = 00010001, zaś jego zwolnieniu — kod 145 = 10010001.

Jakie korzyści płyną z takiego skomplikowania systemu odczytu klawiatury? Otóż komputer PC/XT uzyskuje pełną informację o stanie klawiatury. Mimo, że PC/XT nie ma bezpośredniego dostępu do poszczególnych przycisków, może dokładnie ustalić moment wciśnięcia lub zwolnienia każdego klawisza z osobna. Można np. mierzyć czas wciśnięcia klawisza, można też odpowiednio zareagować w przypadku równoczesnego wciśnięcia kilku klawiszy.

Przypuśćmy np. że program chce ustalić, czy sekwencja jednakowych kodów, wysłanych przez klawiaturę, jest wynikiem wielokrotnego wciśnięcia tego samego klawisza, czy też po prostu dłuższego przytrzymania go w stanie wciśniętym, co uaktywniło funkcję automatycznego powtarzania (autorepeat). Sprawa jest prosta: jeśli po każdym kodzie wciśnięcia następuje kod zwolnienia, to klawisz rzeczywiście był wielokrotnie naciskany. Jeśli kody wciśnięcia są wysyłane jeden za drugim, to zadziałała funkcja autorepeat, a klawisz stale jeszcze jest naciśnięty. W klawiaturze wysyłającej gotowe kody ASCII takie subtelne rozróżnienie nie jest możliwe, chociaż niekiedy mogłoby być przydatne, np. w przypadku, gdy niektórym klawiszom nie należy zezwolić na automatyczne powtarzanie (w klawiaturze PC XT funkcję automatycznego powtarzania mają wszystkie przyciski — włącznie z przyciskami **[Alt]**, **[Shift]** i **[Ctrl]**).

Nie wspomnieliśmy jeszcze, w jaki sposób PC/XT wykorzystuje przesyłanie kodu, informujące o wciśnięciu/zwolnieniu klawiszy. Przesyłane synchronicznie bity są kompletowane i zamieniane na postać równoległą w rejestrze przesuwającym. Po odebraniu ostatniego, ósmego bitu mikroprocesor 8088 otrzymuje sygnał przerwania maskowalnego numer 9. Na przerwanie to reaguje odpowiednia procedura obsługi, zawarta w pamięci ROM, tzw. BIOS (niektóre programy mogą używać własnej procedury obsługi przerwań klawiatury — dotyczy to zwłaszcza gier). Procedura ta odczytuje kod wciśnięcia/zwolnienia z portu nr 96 (060H) i zezwala na odbiór kolejnego kodu. Oprócz tego procedura ta na bieżąco śledzi stan klawiszy **[Shift]**, **[Ctrl]** i **[Alt]**.

PC/XT umożliwia wprowadzanie znaków z klawiatury w jeszcze jeden sposób — przez podanie wprost ich kodu ASCII. Należy w tym celu przycisnąć klawisz **[Alt]**, a następnie — trzymając go wciąż w stanie wciśniętym — wprowadzić jego kod, korzystając z bloku numerycznego po prawej stronie klawiatury (nie z oznaczonych cyframi klawiszy w górnym wierszu klawiatury!). Po zwolnieniu klawisza **[Alt]** zostanie wygenerowany odpowiedni znak. Nie można tylko wysłać w ten sposób znaku NUL (kod = 0). Interpretacja klawiszy bloku numerycznego przy wciśniętym klawiszu **[Alt]** odbywa się jednak już w samym PC/XT, przy udziale procedur w BIOS.

Procedura obsługi przerwania w BIOS interpretuje odpowiednio klawisz i zamienia go na odpowiedni tzw. kod wynikowy. Celowo nie mówimy tu o kodzie ASCII — kod wynikowy niesie bowiem jeszcze dodatkowe informacje. Procedura obsługi uwzględnia przy tym stan klawiszy **[Shift]** i **[Ctrl]** oraz **[CapsLock]** pozwala pisać DUŻYMI LITERAMI bez potrzeby stałego przyciskania

[Shift] i **[Numlock]** (przełącza funkcje klawiszy bloku funkcyjnego: sterowanie kursorem/wprowadzanie danych liczbowych). Kod wynikowy jest wpisywany do specjalnego bufora, mieszczącego do 15 znaków.

Jeśli działający w PC/XT program użytkowy zamierza odczytać znak podany z klawiatury, to oczywiście nie czyni tego, odczytując wprost port nr 96. Wyjaśniliśmy już, że transmitowana za jego pośrednictwem informacja ma specyficzny charakter i może być poprawnie interpretowana jedynie przez procedurę obsługi przerwania. Dla programu użytkowego interesujące są natomiast kody wynikowe, umieszczone w buforze klawiatury, w obszarze RAM służącym za obszar roboczy procedur BIOS. Bezpośredni odczyt zawartości bufora jest co prawda możliwy, lecz wygodniej skorzystać w tym celu z innych procedur, zawartych w pamięci BIOS. Istnieją trzy takie procedury – wszystkie wywoływane są za pośrednictwem przerwania nr 22 (16H; KEYBOARD —IO).

Wszystkie trzy procedury, podobnie zresztą jak inne procedury BIOS, są wywoływane za pośrednictwem rozkazu tzw. przerwania programowego INT (w tym przypadku: INT 16H). W chwili wywołania rejestr AH mikroprocesora 8088 zawiera numer funkcji 0..2. Oto one:

Funkcja 1: odczyt kodu wynikowego kolejnego znaku z bufora BIOS. Odczyt jest niszczący, tzn. następne wywołanie tej funkcji odczyta kod następnego znaku. Jeśli w chwili wywołania procedury bufor klawiatury był pusty, procedura czeka „do skutku", tzn. do chwili wciśnięcia klawisza. W chwili powrotu z przerwania programowego kod rozszerzony jest przekazywany w rejestrze AX.

Funkcja 2: testowanie stanu klawiatury (ściślej: bufora klawiatury). Jeśli bufor jest pusty, flaga procesora ZF przyjmuje wartość 1. Jeśli w buforze dostępny jest przynajmniej jeden znak, to ZF 0, zaś w AX będzie przekazany jego kod wynikowy. Odczyt jest nie niszczący, tzn. następna próba odczytu klawiatury funkcją nr 0 lub 1 dostarczy ponownie kodu tego samego znaku.

Funkcja 3: Odczyt bajtu stanu klawiatury. W chwili powrotu bajt ten znajduje się w rejestrze AL.

Podczas, gdy praktyczne wykorzystanie przedstawionych powyżej funkcji BIOS wymaga znajomości języka asembler, albo przynajmniej wykorzystania odpowiednich mechanizmów dostępu do BIOS w innych językach programowania (np. w TURBO-Pascal), to bajt stanu klawiatury może być interesujący dla wszystkich. Ponieważ w komputerach uchodzących za kompatybilne z PC XT jego lokalizacja w roboczym obszarze BIOS jest ustalona (adres: 1047 ew. 417H), można go odczytywać wprost w języku BASIC (funkcją **PEEK**), TURBO-Pascal (funkcją **MEM**) i większości innych języków programowania. Zawartość bajtu stanu może być w wielu przypadkach niezwykle użyteczna. Oto ona:

Tab. 1

Nr bitu	Znaczenie
7	1 = Tryb wstawiania (**[Insert]**) włączony
6	1 = Aktywny tryb dużych liter (**[CapsLock]**).
5	1 = Aktywny tryb numeryczny (**[NumLock]**)
4	1 = Aktywny klawisz **[ScrollLock]**
3	1 = Klawisz **[Alt]** w stanie wciśniętym.
2	1 = Klawisz **[Ctrl]** w stanie wciśniętym.
1	1 = Lewy klawisz **[Shift]** w stanie wciśniętym.
0	1 = Prawy klawisz **[Shift]** w stanie wciśniętym.

Nietrudno zgadnąć, że bajt stanu jest tym miejscem pamięci operacyjnej, w którym procedura obsługi przerwań klawiatury przechowuje informacje o aktualnym stanie klawiszy funkcyjnych. Jak widać, bajt stanu zawiera wiele danych bardzo użytecznych, ale normalnie niedostępnych. Co ciekawsze, istnieje jeszcze drugi bajt stanu klawiatury o adresie 1048 (418H). Procedury BIOS nie raportują jego stanu — można go odczytać tylko bezpośrednio. W bajcie tym jest wykorzystanych tylko pięć najstarszych bitów:

Tab. 2

Nr bitu	Znaczenie
7	1 = Naciśnięty klawisz **[Insert]**
6	1 = Naciśnięty klawisz **[CapsLock]**
5	1 = Naciśnięty klawisz **[NumLock]**.
4	1 = Naciśnięty klawisz **[ScrollLock]**.
3	1 = Komputer w stanie zatrzymania po **[Ctrl]**+**[Break]**

W odróżnieniu od pierwszego bajtu stanu, cztery najstarsze bity drugiego bajtu informują o aktualnym stanie odpowiednich klawiszy funkcyjnych, a nie wyłącznie o aktualnym stanie danego przełącznika programowego. Dysponując znajomością obu bajtów stanu można więc pokusić się o zaprogramowanie znacznie bardziej zaawansowanych technik wprowadzania danych, niż jest to możliwe na podstawie samego li tylko kodu znaku.

```
;Procedura przykładowa ZamienXY, ilustrująca sposób "przeginania" wektora
;przerwania klawiatury INT16H.              Roland Wacławek, Siemianowice Sl. 86

Zamiana    SEGMENT PARA
           ASSUME  CS:Zamiana

           ORG     100H
Start      JMP     Instal

Int16A:    DD      ?                        ;adres pierwotnej procedury w BIOS

ZamienYZ:  PUSH    BP                       ;przechowaj [illegible] rejestr BP
           MOV     BP,SP                    ;skopiuj zawartość rejestru SP do BP
           CMP     AH,2                     ;test czy kod funkcji w AH 0 lub 1
           PUSHF                            ;przechowaj wynik [illegible] powrotu z BIOS
           PUSH    [BP+6]                   ;skopiuj na stos flagi sprzed INT16H
           CALL CS:DWORD PTR Int16A         ;wywołaj obsługę klawiatury w BIOS
           PUSHF                            ;zapisz flagi na wierzchołku stosu
           POP     [BP+6]                   ;i przenieś je na swoje miejsce
           POPF                             ;odtwórz [illegible] F [illegible] o kodzie funkc[illegible]
           JNC     Koniec                   ;kod funkcji [illegible] 1 [illegible] powr[illegible]
           CMP     AL,'y'                   ;czy wprowadzonym znakiem jest y
           JNE     TestujDY                 ;nie - testuj [illegible] dalej możliwo[illegible]
           MOV     AL,'z'                   ;znak 'y' zamień na znak [illegible]
           JMP     Koniec                   ;natychmiastowy [illegible] z przerwania
TestujDY   CMP     AL,'Y'                   ;czy wprowadzonym znakiem jest 'Y'
           JNE     TestujZ                  ;nie - [illegible] możliwo[illegible]
           MOV     AL,'Z'                   ;znak [illegible] na znak
           JMP     Koniec                   ;natychmiastowy [illegible] z przerwania
TestujZ    CMP     AL,'z'                   ;[illegible]
           JNZ     TestujDZ                 ;nie [illegible] możliwo[illegible]
           MOV     AL,'y'                   ;znak [illegible] zamień [illegible] na znak y'
           JMP     Koniec                   ;natychmiastowy [illegible] przerwania
TestujDZ   CMP     AL,'Z'                   ;czy [illegible] 'Z'
           JNZ     Koniec                   ;natychmiastowy [illegible]
           MOV     AL,'Y'                   ;[illegible] znak [illegible] na znak Y'
Koniec     POP     BP                       ;odtwórz [illegible] wartość BP
           IRET                             ;powrót [illegible] przerwania INT 16H

Instal     MOV     AH,35H                   ;funkcja DOS: [illegible] przerwania
           MOV     AL,16H                   ;numer przerwania obsługi klawiatury
           INT     21H                      ;wołaj DOS, powrót z wektorem w ES:BX
           MOV     WORD PTR Int16A,BX       ;zapamiętaj pierwotny wektor przerwa-
           MOV     WORD PTR Int16A+2,ES     ;nia, [illegible] procedurę w BIOS
           MOV     DX,OFFSET ZamienYZ       ;[illegible] procedury ZamienYZ w DX
           MOV     AH,25H                   ;funkcja DOS: ustaw wektor przerwania
           MOV     AL,16H                   ;numer przerwania obsługi klawiatury
           INT     21H                      ;wołaj DOS, ustaw wektor dany w DS:DX
           MOV     DX,OFFSET Instal         ;przesunięcie [illegible] pierwszy wolny bajt
           INT     27H                      ;uczyń procedurę ZamienYZ rezydującą

Zamiana    ENDS
           END     Start
```

Wracając do wynikowego kodu znaku: obejmuje on dwa bajty, przekazywane w chwili powrotu z procedury BIOS KEYBOARD-IO w starszej (**AH**) i młodszej (**AL**) części rejestru **AX**. Bajt w **AL** jest po prostu kodem ASCII, odpowiadającym wciśniętemu klawiszowi – o ile z danym klawiszem taki znak jest związany; w przeciwnym razie **AL** zawiera **0**. Rejestr **AH** zawiera natomiast znany nam już skądinąd numer klawisza. Po co? Dla odróżnienia znaków ASCII, wprowadzonych pojedynczym

przyciśnięciem klawisza, od znaków o tych samych kodach, ale podanych za pośrednictwem klawisza **[Alt]** i bloku klawiszy numerycznych. W tym ostatnim przypadku rejestr **AH** zawiera **O**, co nie odpowiada numerowi żadnego z istniejących klawiszy.

Jeśli wciśnięto klawisz nie mający odpowiednika w zbiorze znaków ASCII, np. **[Insert]** lub **[F1]...[F10]**, to rejestr **AL** przyjmie wartość 0, natomiast w **AH** zostanie przekazany tzw. rozszerzony kod klawisza. W większości przypadków chodzi tu zresztą o kody generowane przez kombinacje klawiszy, np. zapis: [Alt] + [A] należy odczytać jako równoczesne wciśnięcie klawiszy: [Alt] oraz [A]. Kody odpowiadające poszczególnym klawiszom lub ich kombinacjom są przedstawione w poniższej tablicy:

Tab. 3

Kod szesn	Kod dzies.	Klawisze lub ich kombinacje
3	3	[Ctrl]+[C]
F	15	[Ctrl]+[Tab]
10..16	16..22	[Alt]+[Q], [W], [E], [R], [T], [Y], [U]
17..19	23..25	[Alt]+[I], [O], [P]
1E 24	30..36	[Alt]+[A], [S], [D], [F], [G], [H], [J]
25..26	37..38	[Alt]+[K], [L]
2C .32	44..50	[Alt]+[Z], [X], [C], [V], [B], [N], [M]
3B..44	59..68	[F1] .[F10]
47	71	[Home]
48	72	[Kursor w gorę]
49	73	[PgUp]
4B	75	[Kursor w lewo]
4D	77	[Kursor w prawo]
4F	59	[End]
50	80	[Kursor w dół]
51	81	[PgDn]
52	82	[Ins]
53	83	[Del]
54..5D	84..93	[Shift]+[F1]..[F10]
5E..67	94..103	[Ctrl]+[F1]..[F10]
68..71	104..113	[Alt]+[F1]..[F10]
72	114	[Ctrl]+[Prtsc]
73	115	[Ctrl]+[Kursor w lewo]
74	116	[Ctrl]+[Kursor w prawo]
75	117	[Ctrl]+[End]
76	118	[Ctrl]+[PgDn]
77	119	[Ctrl]+[Home]
78..81	120..129	[Alt]+[1]..[9]
82	130	[Alt]+[0]
83	131	[Alt]+[ ]
84	132	[Ctrl]+[PgUp]

Jak widać, niektóre kombinacje klawiszy, np. **[Alt]** **[Spacja]**, są ignorowane. Odpowiedzialna za to jest jednak już procedura obsługi przerwania w BIOS. Klawiatura wysyła w każdym razie wszystkie niezbędne informacje. Programy, które muszą korzystać z niestandardowych kombinacji klawiszy, instalują często własne precedury obsługi przerwania nr 9. Jest to dość proste, gdyż wektory wszystkich przerwań są zapisane w pierwszym kilobajcie pamięci RAM (każdy z 256 możliwych wektorów zajmuje po 4 bajty).

Znacznie częściej wykorzystywaną możliwością jest zmiana wektora przerwania nr 16H, poprzez który odczytuje stan klawiatury większość programów. Przeważnie nie rezygnuje się przy tym całkowicie z usług procedur w BIOS, lecz dołącza do nich własny program maszynowy, funkcjonujący na podobieństwo „filtru". Można w ten sposób praktycznie dowolnie przedefiniować całą klawiaturę, wzbogacając ją np. o możliwość bezpośredniego wprowadzania znaków graficznych lub — w powiązaniu z innymi programami — polskich liter (ą, ó itd.). Samodzielne „przegięcie" wektora przerwania wymaga jednak z reguły znajomości podstaw programowania w języku asembler procesora 8088 86.

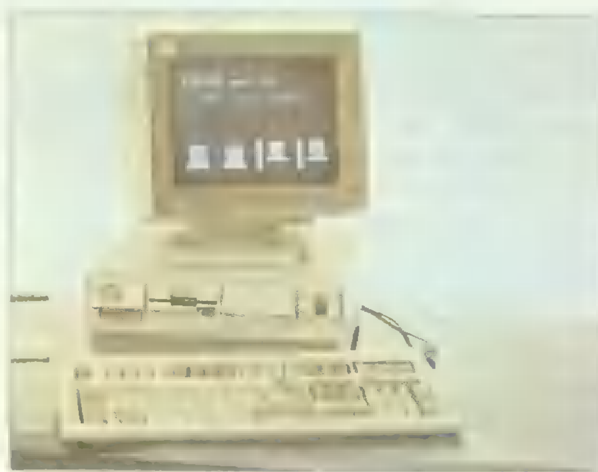
Bardziej zaawansowanych użytkowników IBM-PC, zainteresowanych własnymi rozszerzeniami systemu, zainteresuje być może przykładowy program w języku asembler, ilustrujący ogólną zasadę „przeginania" wektora INT 16H (listing 1). Właściwa funkcja programu jest banalna i polega na zamianie miejscami liter Y i Z. Może to być przydatne dla osób, przyzwyczajonych do pracy z klawiaturą typu niemieckiego (QWERTZ), i zmuszonych do korzystania z najczęściej dostarczaną klawiaturą angielską (QWERTY).

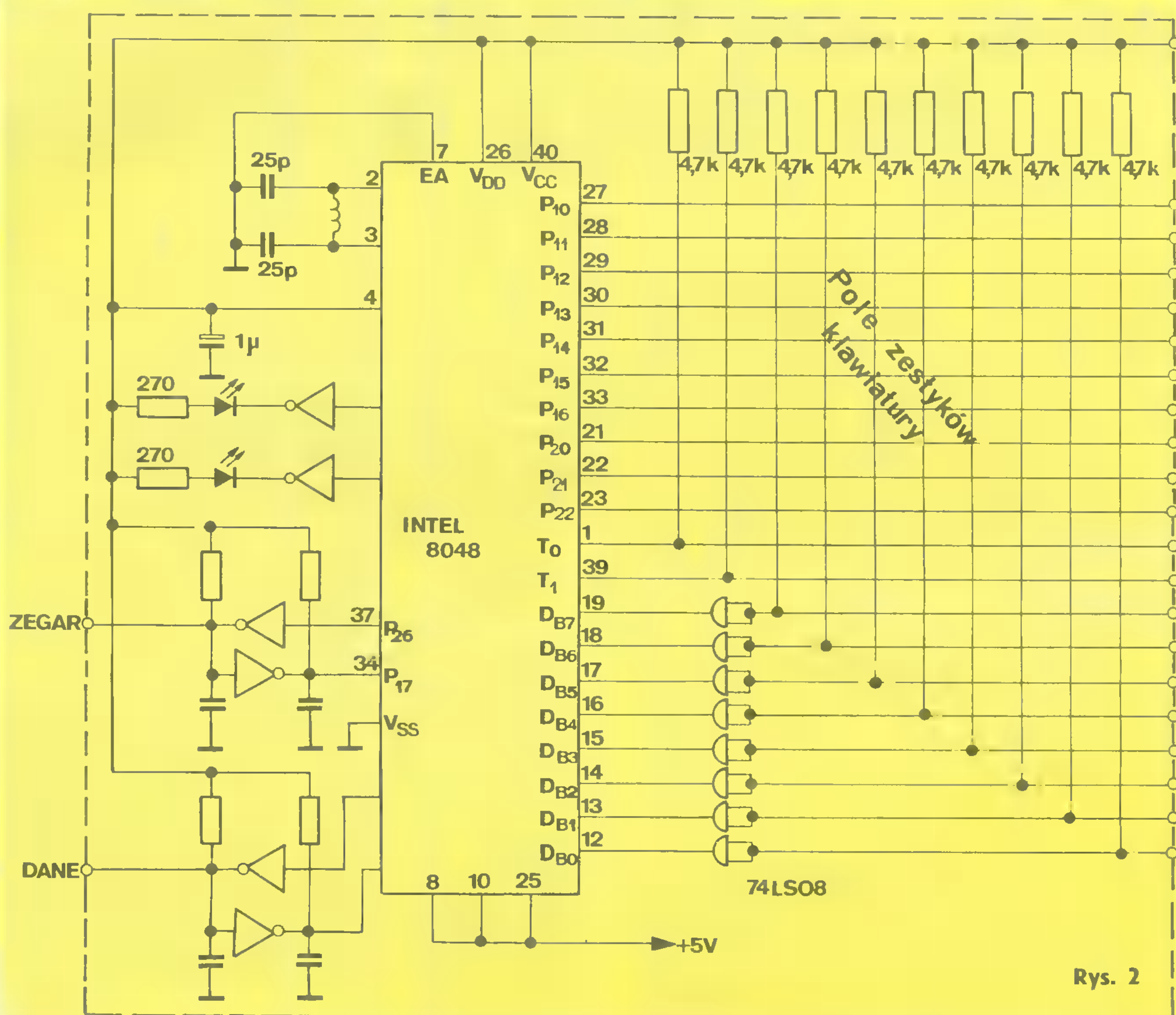
Po zapisaniu tekstu programu na dysku za pomocą edytora, plik źródłowy (nazwany np. YZ) należy poddać asemblacji za pomocą asemblera np. MASM, a następnie łączeniu za pomocą linkera **LINK**. Uzyskiwany jest program wynikowy **YZ.EXE**. Przed uruchomieniem należy przekształcić go jeszcze na format .COM za pomocą programu EXE2BIN:

EXE2BIN YZ.EXE YZ.COM

Program składa się z dwóch części: filtra, włączanego w procedurę obsługi przerwania (ZamienYZ) oraz procedury instalacyjnej (Instal). Aby program podjął działanie, wystarczy jego jednokrotne uruchomienie. Po uruchomieniu z poziomu systemu PC-DOS/MS-DOS zleceniem: YZ, sterowanie zostanie przekazane do procedury instalacyjnej **Instal**. Procedura ta odczyta stary wektor przerwania INT 16H (wskazujący procedurę obsługi w BIOS), zapamięta stary wektor w komórce Int16A, po czym wstawi nowy wektor przerwania INT 16H, wskazujący tym razem bezpośrednio adres **ZamienYZ**. Następnie procedura Instal przekaże sterowanie do systemu DOS w ten sposób, aby procedura **ZamienXY** została włączona w obszar systemowy, chroniony przed nałożeniem programów użytkowych.

Po każdym wywołaniu przerwania programowego INT16H uruchamiana jest procedura **ZamienYZ**. Jej zadanie jest nieco trudniejsze, niż w przypadku przeginania innych wektorów, bowiem po powrocie z przerwania INT16H dla funkcji 1 istotny jest bit Z w rejestrze stanu. Szkopuł w tym, że po przerwaniu programowym INT16H, jak zresztą po każdym przerwaniu w procesorze





Rys. 2

8088 86, rejestr stanu procesora jest automatycznie zapisywany na stos. Rozkaz powrotu z przerwania **IRET** odtwarza z kolei automatycznie bity stanu sprzed przerwania.

Aby zneutralizować wpływ „przymusowej" ochrony bitów stanu, wystarczy prosta sztuczka. Wystarczy odczytać ze stosu zapamiętaną tam zawartość rejestru stanu sprzed przerwania i skopiować ją ponownie na stos, w miejsce, w którym umieściłby ją kolejny rozkaz przerwania programowego **INZ**. Następnie wystarczy wykonać rozkaz **CALL** pod zapamiętany w chwili instalacji adres w BIOS-ROM. Efekt wykonania pary rozkazów: **PUSH [BP + 6]** i **CALL** jest taki sam, jak rozkazu **INT**, z wyjątkiem tego, że nie następuje automatyczne blokowanie przerwań maskowalnych. Przedtem należy jednak sprawdzić, czy numer funkcji wynosi **0** lub **1** (w przypadku funkcji **2** zamiana kodu nie jest potrzebna) i zapisać tę informację na stosie.

Po powrocie z procedury obsługi w BIOS wystarczy skopiować aktualną zawartość rejestru stanu na stos, w miejsce, z którego poprzednio została odczytana. Dzięki temu po kolejnym rozkazie **IRET**, kończącym procedurę **ZamienYZ**, zawartość rejestru stanu będzie identyczna, jak w chwili powrotu z BIOS. Dalsze czynności procedury nie wymagają komentarza — trzeba sprawdzić, czy kod znaku w **AL** nie jest zgodny z kodem którejś z liter: x, z, X, Z, a jeśli tak, zastąpić go właściwym odpowiednikiem. W razie potrzeby zestaw realizowanych funkcji można oczywiście znacznie rozbudować.

Przyjęty w PC XT system raportowania stanu klawiatury pozwala wprowadzić za jej pośrednictwem znacznie bogatszy zestaw informacji, niż w przypadku konwencjonalnych klawiatur autonomicznych, bez rezygnacji z zalet klawiatury luźno związanej z komputerem przy pomocy długiego przewodu lub nawet — jak np. w komputerze PC-Junior — transmisją w podczerwieni. Możliwość konstruowania i integrowania z systemem operacyjnym, a nawet z BIOS, własnych procedur obsługi, zapewnia z kolei niezbędną elastyczność i łatwość adaptacji komputera do różnych warunków.

JAK ZMIEŚCIĆ JESZCZE WIĘCEJ?

Tytuł artykułu mógłby skojarzyć się przeciętnemu Polakowi z niebywałym tłokiem w autobusach i innych środkach komunikacji: pasażerowie nierzadko dwoją się i troją, jakby tu zmieścić w ciasnym wnętrzu jeszcze choć kilka osób... Tym razem jednak nie będziemy pisać o wyczynach typu podróż nad morze w lipcu, ale o masowych pamięciach zewnętrznych mikrokomputerów. Tam też panuje ogromny tłok, choć jego zwiększanie polega tym razem na ograniczaniu miejsca zajmowanego przez jednostkę informacji. Cel jest oczywisty – nieduża gabarytowo pamięć jest z pewnością wygodniejsza oraz tańsza od większej, bardzo często jest też szybsza.

W wielu zastosowaniach najkorzystniejszym rodzajem pamięci zewnętrznej o dużej pojemności jest magnetyczna pamięć kinetyczna. Przykładem pamięci tego rodzaju są bardzo rzadko już dziś stosowane pamięci taśmowe czy kasetowe oraz będące w powszechnym użyciu pamięci na dyskach elastycznych i na dyskach twardych. Największe postępy w zakresie „zagęszczania" informacji poczyniono w pamięciach na dyskach twardych niewymiennych, określanych w większości przypadków mianem Winchester (dla mikrokomputerów). Stosuje się cienkowarstwową technologię warstw magnetycznych, obniża się do minimum wysokość „lotu" głowicy nad talerzem dysku (głowice poruszają się na mikronowej poduszce powietrznej), zmniejsza się odległości między ścieżkami, wprowadza dynamiczne śledzenie śladu (konieczne przy małej szerokości ścieżki) i wiele innych środków, które ogólnie biorąc składają się na precyzję układów mechanicznych i sterujących ruchem głowic. To jednak nie jest jedyny sposób zwiększania gęstości informacji — duże rezerwy tkwią w sposobie kodowania informacji. Zanim jednak podamy kilka szczegółów o nowej metodzie kodowania, przypomnijmy pokrótce podstawowe, stosowane obecnie metody zapisu danych na nośnikach magnetycznych. Przy okazji zapoznamy się również z kilkoma specyficznymi dla tego zapisu wymaganiami.

Zewnętrznie Winchester systemu RLL niczym się nie różni od poprzedników MFM — na zdjęciu model Miniscribe 8425 (3 1/2 cala, 4 głowice, 612 cylindrów, pobór mocy — 11 W). Nawiasem mówiąc wiele Winchesterów przystosowanych fabrycznie do MFM może być wykorzystanych z kontrolerem RLL bez dokonywania jakichkolwiek przeróbek...



Nośnikiem informacji w pamięciach magnetycznych jest odpowiednia warstwa o własnościach ferromagnetycznych, a konkretnie stan jej namagnesowania w charakterystycznych punktach. Wyróżnić możemy dwa stany namagnesowania: dodatni i ujemny. Odpowiada to w pewnym przybliżeniu zeru i jedynce w dwójkowym systemie liczenia. W tym momencie można by zadać pytanie: dlaczego nie wykorzystujemy stanów pośrednich, czyli np. zerowego namagnesowania, połówkowego itp. To jednak tak, jakbyśmy chcieli wprowadzić logikę tzw. dwu-, trój- i więcej wartościową, np. napięciu od 0 do 1 V odpowiada 0, od 1 do 2 V — 1 itd. — konsekwencją byłoby znaczne skomplikowanie układów elektronicznych, zmniejszenie szybkości pracy i znacznie większe prawdopodobieństwo przekłamania. Dodatkową trudnością przy nośniku magnetycznym jest trudność rozróżnienia znaku namagnesowania nośnika przy powolnych jego zmianach. Pełny odczyt natężenia pola magnetycznego, z dokładnością do jego składowej stałej jest możliwy ponadto obecnie jedynie za pomocą czujników hallotronowych, które jednak zupełnie nie nadają się do pracy w szybkich pamięciach z powodu niskiej częstotliwości granicznej. Pozostają zatem jedynie głowice indukcyjne, które z kolei reagują jedynie na zmianę kierunku namagnesowania (zmianę znaku). Przy niedużych częstotliwościach można wprowadzić wzmacniacz całkujący i w pewnym zakresie realne jest odtworzenie rzeczywistego przebiegu namagnesowania. Dla większych szybkości jest to już jednak bardzo trudne i niecelowe — aby uniknąć owych trudności i poprawić parametry zapisu, w charakterze jednostki informacji przyjęto zmianę kierunku namagnesowania nośnika.

Mogłoby się wydawać, że przyjęcie w charakterze informacji zmiany kierunku namagnesowania nośnika nie wprowadza żadnych ograniczeń. Mamy ciąg zer — nie ma impulsu z głowicy (świadczyłby on o zmianie namagnesowania). Pojawia się jedynka — jest impuls itd. Niestety — przy długich ciągach jednakowych bitów nie odczytujemy żadnych impulsów, nie ma więc mowy o jakiejkolwiek synchronizacji (nie wiemy, kiedy spodziewać się impulsu odpowiadającego n-temu bitowi). Dodajmy, że synchronizacja za pomocą pomiaru obrotów jest tu praktycznie nierealna, gdyż jednemu bitowi odpowiada nie więcej niż kilka mikrometrów na nośniku — jest to znacznie mniej, niż możliwa w tym przypadku rozdzielczość odpowiednich układów dekodujących położenie. Trzeba zatem wprowadzić dodatkową ścieżkę synchronizacji (wykorzystuje się ją w zapisie NRZI, stosowanym w pamięciach taśmowych — dzisiaj już

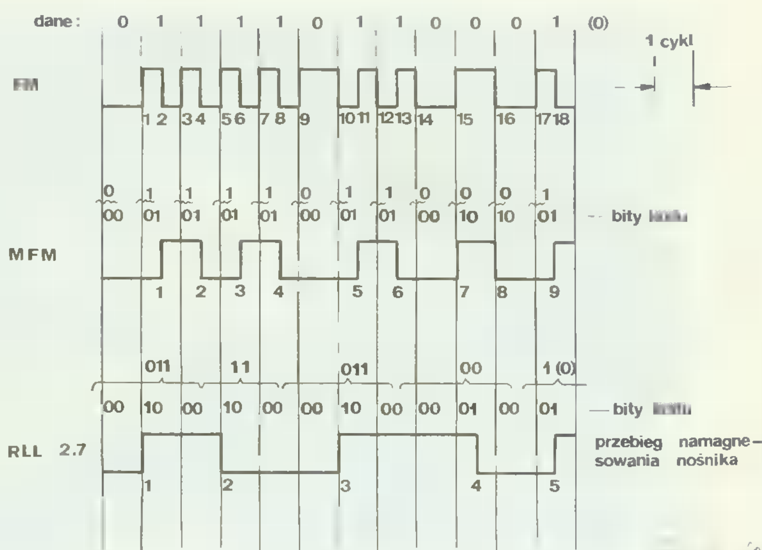
o małym znaczeniu) lub dodatkowe impulsy umożliwiające układowi elektronicznemu właściwe zsynchronizowanie się z odczytywanym sygnałem. Przykładem systemu kodowania danych mającego tę właściwość jest modulacja FM (podobne przebiegi uzyskuje się dla stosowanej w pamięciach kasetowych modulacji PE — kodowania fazowego) (rys. 1).

Przed dokładnym objaśnieniem przebiegów z rys. 1 musimy wprowadzić pojęcie tzw. bitu kodu. W odróżnieniu od bitu informacji (czyli tego, co interesuje użytkownika), bit kodu określa przebieg namagnesowania nośnika. Jego znaczenie jest dość oczywiste — informacja na nośniku przekazywana jest przez zmiany kierunku namagnesowania w określonych chwilach — a zatem w chwilach tych bit kodu przyjmuje wartość jeden. Jeżeli nie występuje zmiana kierunku namagnesowania, bit kodu jest równy zeru. Wszystko to musi być odpowiednio zsynchronizowane z sygnałem zegarowym, zapewniającym zachowanie właściwych zależności czasowych i bezbłędny odczyt.

Spojrzmy teraz na rys. 1. Na górze są wypisane bity informacyjne (danych), odstęp między dwiema pionowymi liniami odpowiada jednemu cyklowi zegarowemu. Pierwszy od góry przebieg odpowiada modulacji częstotliwości (FM — *Frequency Modulation*). Zauważamy, że na początku każdego cyklu zegarowego następuje zmiana kierunku namagnesowania — odpowiadający bit kodu ma wartość jeden. W środku każdego cyklu zmiana kierunku namagnesowania występuje tylko wtedy, gdy bit informacji równy jest jeden. Na jeden bit informacyjny przypadają zatem dwa bity kodu: pierwszy jest zawsze równy jeden, drugi ma wartość bitu informacji (danych kodowanych w ten sposób).

Jak widać na rysunku, dwunastu bitom informacji przypada w modulacji FM aż 18 przemagnesowań nośnika. Oznacza to dużą częstotliwość pracy układów zapisu i odczytu, a zatem przy ograniczeniu jej od góry osiągana szybkość zapisu jest stosunkowo niewielka, podobnie też jego gęstość. Dążąc do poprawienia parametrów kodu opracowano zmodyfikowaną modulację FM, oznaczoną MFM (*Modified Frequency Modulation*) — na rys. 1 odpowiada jej drugi przebieg. Widać wyraźnie, że ilość przemagnesowań jest znacznie mniejsza — dwukrotnie w stosunku do FM. Proces kodowania danych w metodzie MFM jest zbliżony do FM — (tab. 1). Różnica dotyczy pierwszego bitu kodu w danym cyklu — dla FM był on zawsze równy jeden, w MFM jest równy negacji poprzedniego bitu danych. W taki sposób zapewniono, że przynajmniej raz na dwa cykle pojawi się jedno przemagnesowanie, umożliwiające właściwą synchronizację odczytu danych.

System MFM jest obecnie stosowany bardzo szeroko — oferowanych jest wiele scalonych kontrolerów pracujących w ten sposób. Najszerszą domeną metody MFM są dyski elastyczne (dyskietki), a do niedawna były też dyski twarde, w tym tzw. Winchestery. Zastosowany sposób kodowania należy do bardzo szerokiej grupy kodów, opracowanych teoretycznie w roku 1975 przez pracownika koncernu IBM, A. M. Patela. Patel wprowadził nowy sposób opisu różnych metod kodowania, które w skrócie można przedstawić, jako KOD — F (d, k, m, n, r)



Rys. 1. Porównanie zapisów FM, MFM oraz RLL 2,7 — widać wyraźnie, że kod RLL 2,7 wywołuje znacznie mniej zmian namagnesowania nośnika

Znaczenie poszczególnych parametrów jest następujące:

d — minimalna ilość zer pomiędzy dwiema jedynkami (oczywiście bitów kodu),

k — maksymalna ilość zer pomiędzy dwiema jedynkami bitów kodu,

m — ilość bitów danych, które są zamieniane w n bitów kodu.

Znaczenie parametru r nie jest już tak łatwe do wytłumaczenia. W pewnym przybliżeniu określa on czy bit danych jest zamieniany na stałą ilość bitów kodu, czy też w zależności od kombinacji bitów danych są generowane różne ciągi bitów kodu. W drugim przypadku mamy do czynienia z tzw. kodami RLL (*run-length-limited*).

Ciekawostką jest to, że metoda MFM także należy do kodów typu RLL — można ją opisać parametrami (1, 3, 2, 1). Natomiast stosowana obecnie metoda RLL ma parametry (2, 7, 1, 2, 3) — stąd w skrócie nazywa się ją RLL 2, 7.

Uzyskiwany w metodzie RLL ciąg bitów kodu jest zależny od kombinacji bitów danych (tab. 1). Jest to istotne utrudnienie przy kodowaniu, a szczególnie

Tabela 1
Porównanie sposobów kodowania w metodach FM, MFM i RLL2,7

Metoda	Dane	Bity kodu	Uwagi
FM	0 1	10 11	
MFM	0 1	x0 01	x — negacja poprzedniego bitu danych
RLL2,7	000 10 010 0010 11 011 0011	000100 0100 100100 00100100 1000 001000 00001000	

Tabela 2

Porownanie parametrow kodow MFM i RLL2,7

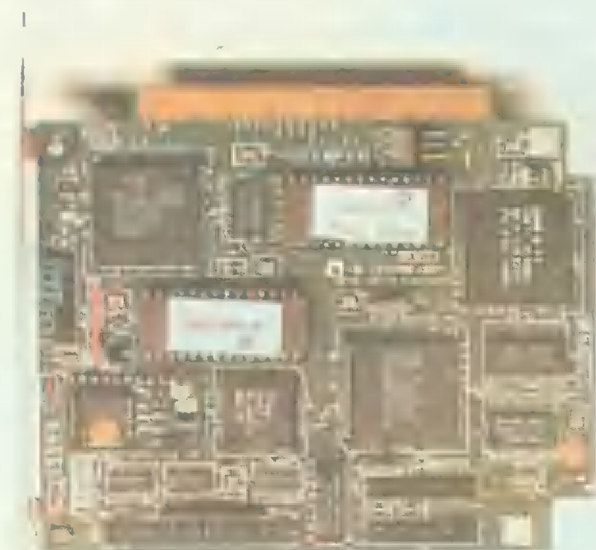
Kod	parametr							
	d	k	m	n	r	R	F	D
MFM	1	3	1	2	1	0,5	2,0	1,0
RLL2,7	2	7	1	2	3	0,5	2,67	1,5

dekodowaniu informacji, ale za to kod ten jest bardzo efektywny. Z analizy przebiegu na rys. 1 jasno wynika, że metoda RLL 2,7 redukuje o ok. 35 procent ilość przemagnesowań nośnika — można zatem, przy niezmienionej maksymalnej częstotliwości pracy, półtorakrotnie zwiększyć gęstość zapisu danych.

Spróbujmy porównać ze sobą metody MFM oraz RLL 2,7. Wprowadzono w tym celu kilka parametrów charakteryzujących użytkowe właściwości tychże systemów kodowania — przedstawione są one w tab. 2. Parametr R (zwany z ang. *Code Rate*) określa stosunek ilości bitów danych do ilości bitów kodu — w naszym przypadku dla obu metod jest on równy 0, 5, czyli jednemu bitowi informacji odpowiadają dwa bity kodu.

Parametr F odgrywa w naszym porównaniu bardzo istotną rolę — określa on mianowicie zakres częstotliwości, w którym mieści się zakodowany sygnał. Konkretnie F równe jest stosunkowi maksymalnej i minimalnej częstotliwości zakodowanego sygnału (częstotliwości mierzonej, jako wartość chwilowa, czyli odwrotność okresu). Jest oczywiste, że dla mniejszej wartości tego parametru pasmo układów zapisu i odczytu jest węższe, czyli łatwiejsze jest spełnienie określonych parametrów jakościowych przebiegów otrzymywanych w układach zapis/odczyt. W tym momencie widzimy jak zwiększono gęstość zapisu — pasmo dla RLL 2,7 jest o jedną trzecią szersze niż dla MFM.

Kontroler RLL mało znanej firmy SMS OMTI zajmuje jedynie połówkowy pakiet formatu IBM PC. Jest to drugi po Adaptecu ACB 2070 A (duży pakiet) kontroler dysku twardego wykorzystujący kodowanie RLL, jednak pierwszy w małym formacie. Oczywiście Adaptec też nie spał — kilka miesięcy później powstał model ACB-2071



Parametr D jest miarą efektywności kodu i określa względną ilość bitów danych przypadających na jedno przemagnesowanie nośnika — widać, że metoda RLL 2,7 jest o 50 procent efektywniejsza od MFM. Stąd właśnie bierze się fakt, że starsze modele dysków Winchester po zastosowaniu nowej metody zapisu zyskują o połowę większą pojemność (np. zamiast 20—30 MB). Teoretycznie mogłoby się wydawać, że każdy model przystosowany do metody MFM mógłby pracować także z RLL z pojemnością większą o 50 procent. W rzeczywistości nie jest to jednak tak łatwe.

Dlaczego nie wszystkie urządzenia przystosowane do metody MFM mogą pracować w RLL? Spójrzmy na tabelę porównawczą i różnice między MFM i RLL. Na pierwszym miejscu widać większy zakres częstotliwości. Stwarza to bardziej krytyczne zależności w torach wzmacniających, konieczne jest też zachowanie odpowiedniej charakterystyki amplitudowo-fazowej w znacznie szerszym paśmie roboczym. Wiadomo, że przebiegi w głowicach pracujących przy tak wysokich częstotliwościach (powyżej 5 MHz) mają już charakter rezonansowy i wypadkowa charakterystyka jest typu pasmowego. Uzyskanie większej szerokości pasma wymaga zastosowania lepszej korekcji w torach wzmacniających, konieczna może też być selekcja głowic. Dużym utrudnieniem przy odczycie jest fakt występowania znacznie mniejszej ilości impulsów synchronizujących w stosunku do liczby bitów informacji, pogarszający bardzo warunki pracy układów synchronizacji. W przypadku MFM synchronizacja ta jest stosunkowo łatwa, wykorzystuje się pętlę synchronizacji fazowej (PLL), natomiast w RLL konieczna jest już znacznie większa stabilność pracy, bezpośrednio wpływająca na możliwość przekłamań. W metodzie RLL może występować do 7 zer między jedynkami bitów kodu, a w MFM jedynie do 3 — a zatem wymagania stabilności rosną ponad dwukrotnie.

Większa efektywność kodu RLL nie wiąże się z podwyższeniem maksymalnej częstotliwości zapisu, ale za to maleją 1,5 raza szerokości „okien" czasowych, w których należy testować wystąpienie jedynek kodu. O ile w MFM przedział ten ma szerokość 100 ns, to przy RLL już tylko 67 ns — stwarza to oczywiście znacznie większe trudności przy odczycie — wymagana jest znacznie większa szybkość pracy układów dekodujących. Mniejsza szerokość okna to także mniejszy margines zakłóceń — wzrasta więc niekorzystny wpływ szumów nośnika, głowicy, układów elektronicznych, a także niestabilności obrotów i układów pozycjonowania głowic. Reasumując — urządzenie przystosowane do pracy w systemie MFM musi charakteryzować się bardzo wysoką jakością oraz pewnym zapasem w zakresie parametrów decydujących o dokładności i jakości odczytu, aby możliwe było jego wykorzystanie w systemie RLL z o połowę większą pojemnością.

Obecnie już niemal wszystkie Winchestery renomowanych firm są przystosowane do pracy w systemie RLL. Opracowano także już scalone wersje kontrolerów, zajmujących zaledwie jedną krótką kartę formatu IBM PC. Kontrolery te stosują kody korekcyjne, umożliwiające nie tylko bardzo niezawodną detekcję błędów, gwarantującą niesłychanie małe prawdopodobieństwo przekłamania (wykrywane są błędy o długości do 27 bitów),





Mini-Winchester instalowany w miejscu standardowego pakietu IBM-owskiego, zwany HARDCARD. Zauważmy, że w połowie jest to... kontroler na połowkowym pakiecie. Druga połowa to mini-Winchester 3 1 2 cala. W hard-karcie stosuje się już technologie wykorzystywane dotychczas jedynie w urządzeniach o bardzo wysokiej niezawodności. Na zdjęciu model firmy Plus Development Corporation (ze zdjętą obudową). Krótkie dane techniczne: pojemnosc — 20 MB; sredni czas dostępu — 49 ms; pobierana moc — [illegible] W; niezawodnosc — 40 000 godzin (!); zapis — RLL 2,7

ale także korekcje błędów o długości do 11 bitów (dane dla kontrolera 5527, korekcja 48-bitowa). Należy oczekiwać, że w najbliższym czasie metoda RLL stanie się podstawową w masowych pamięciach magnetycznych, po przełamaniu pewnych trudności związanych z występującą czasem niekompatybilnością z wzorcem IBM.

Na zakończenie przykład miniaturyzacji na miarę dwudziestego wieku. Okazuje się, że pamięć 20 lub 30 MB na dysku typu Winchester wcale nie musi zajmować tak dużej ilości miejsca, jak napęd dyskietki 5 1 4 cala — mieści się swobodnie na jednej dużej karcie (Winchester 3 1 2 cala) wkładanej do pierwszego slotu (gniazda na głównej płycie IMB-a). Stosuje się oczywiście kodowanie RLL, dostępne modele posiadają pojemność 20 lub 30 MB, na tej samej płycie znajduje się także kontroler. Ciekawe, kiedy dojdziemy do granic tej miniaturyzacji? Wszak nie można wszystkiego upychać bez końca... Prawdopodobnie w chwili ukazania się niniejszego numeru dostępne będą już modele o większej pojemności, być może opracuje się też nowy, efektywniejszy sposób kodowania.

Na podstawie „PC Magazin" opracował
(g.z.)

POMIĘDZY SINCLAIREM A COMMODOREM

Część II

W numerze „MT" 9/86 opisaliśmy sposób dołączenia do najpopularniejszego w Polsce mikrokomputera ZX Spectrum peryferiów Commodore'a — jednych z najtańszych na rynku. Konkretnie był opisany odpowiedni interfejs oraz handler plotera VC1520. Obiecano wtedy, że w następnych odcinkach będą opisane handlery drukarek MPS801 i MPS803 — tutaj wkradło się jednak pewne opóźnienie, niezawinione przez redakcję. Lepiej jednak późno, niż wcale, więc w tym artykule opiszemy najbardziej chyba poszukiwany handler do drukarek MPS801 i 803.

Podstawową częścią programu obsługującego wspomniane drukarki jest handler magistrali IEC625, prawie identyczny z opisanym w numerze 9/86. Zmiany dotyczą jedynie dodania testu BREAK, bardzo ważnego w przypadku „zawieszenia" drukarki czy konieczności szybkiego przerwania druku oraz zmiany nazwy ostatniej komórki wykorzystywanej przez program. Pełny handler drukarki, łącznie z handlerem magistrali IEC625, przedstawia listing 1.

Nie będziemy tutaj analizować obsługi magistrali (linie od 1800 w górę), zapoznamy się tylko z funkcjami programu obsługującego odpowiednie przekodowanie znaków i realizowanie niektórych funkcji druku. Zaczynając od początku, natrafiamy na procedurę inicjalizacji handlera. Polega ona na wpisaniu do tablicy adresów kanałów we wy komputera (jej początek określa zmienna systemowa CHANS) odpowiedniego adresu obsługi drukarki (kanał 3, stąd poprawka 3 * 5 = 15 — linia 120, 130). Ponadto należy wyzerować wszystkie komórki robocze oraz rejestry interfejsu (linia 230).

Wejście do programu znajduje się w linii 270 (etykieta MPS803). Na początku następuje rozpoznanie stanu drukarki i rodzaju jej pracy. Bit 7 zmiennej STATFL określa, czy była zainicjowana transmisja — inicjuje się ją dopiero na początku druku znaków i każdorazowo kończy po wysłaniu CR tak, aby można było m.in. korzystać z ręcznego wysuwu papieru. Następnie porównujemy znak wysyłany do drukarki (znajduje się on w akumulatorze) z ØFH — jest to znak końca trybu druku graficznego. Bit 4 zmiennej STATFL określa, czy jest włączony ów tryb druku graficznego — jeżeli tak, to nie należy dokonywać przekodowywania zawartości akumulatora. Bit 6 i 5 wskazują na pobieranie danych funkcji TAB — pierwszy bajt jest ładowany do zmiennej TABBUF, drugi (wartość ponad 255) zaniedbujemy.

Jeżeli nie ma funkcji TAB, przechodzimy do linii 520 (WEJ). Sprawdzamy, czy znak nie jest znakiem sterującym (< 20H). Następnie należy przekodować kody znaków odpowiednio do wymagań drukarki — najpierw sprawdzamy, czy wysyłany kod jest mniejszy od 41H (nie są to jeszcze litery). Jeżeli tak, nie jest wymagana korekcja kodu. Dla kodów od 41H do 5AH należy dodać poprawkę 20H. Jeżeli kod jest większy od 5AH, następuje rozpoznanie kilku znaków w podprogramie ROZPOZ (linie 1470—1620). Podprogram korzysta z tablicy wzorców i kodów wynikowych (etykieta TABZN) — kody są grupowane parami, najpierw wzorzec, potem odpowiednik dla drukarki. W przypadku znalezienia wzorca do akumulatora jest ładowany odpowiedni kod z tablicy. Jeżeli dany znak nie występuje w zestawie drukarki, jest wstawiany kod spacji. Po wykonaniu podprogramu wskaźnik C (*Carry*) informuje o znalezieniu znaku w tablicy lub o braku konieczności przekodowania innych znaków o kodach poniżej 61H. Dla ustawionego C następuje zatem druk znaku.

Dla innych kodow należy dokonać dalszego rozpoznania (linia 640). Kody mniejsze od 7BH należą do małych liter — wtedy należy odjąć poprawkę 20H. Dla kodow pozostałych przechodzimy do linii 680. Teraz sprawdzamy, czy wysyłany kod nie jest jednym ze sterujących (czyli 91H lub 92H) oraz czy nie jest kodem nazwy funkcji w Spectrum (są to kody od A5H wzwyż). Pozostałe kody odpowiadające grafice użytkownika oraz semigrafice są zamieniane na spacje.

W przypadku wykrycia kodu nazwy, następuje jej druk (linia 740—920). Ten fragment programu jest praktycznie przepisany z ROM-u — korzysta on z oryginalnej procedury przeszukiwania tablicy nazw, następnie drukuje znaki z tablicy aż do napotkania znaku z ustawionym najstarszym bitem (przy druku bit ten jest zerowany — linia 790). Na końcu tej procedury jest w razie konieczności, dodrukowywana spacja.

Dalszy ciąg programu rozpoznaje kody sterujące, rozpoczynając od kodu trybu graficznego (czyli 8). Po jego wykryciu ustawiony zostaje odpowiedni wskaźnik (bit 4 STATFL) oraz zerowany jest pomocniczy bufor tabulacji TABBF2. Następnie rozpoznaje się kod wyłączenia trybu graficznego (OFH) — linia 1030. Po jego wyłączeniu przeprowadza się korekcję wskaźnika POZYCE, określającego aktualną pozycję druku (wykorzystywana przy TAB i PRINT z przecinkiem). Każdorazowo w pętli odejmuje się 6 (ilość punktów na jeden znak) i odpowiednio dodaje 1 do POZYCE aż do wyzerowania lub negacji akumulatora. Korekcja ta jest wymagana dla prawidłowego działania funkcji TAB i „PRINT", przy np. drukowaniu polskich liter w trybie graficznym.

Ciąg dalszy rozpoznawania znakow sterujących następuje w linii 1150. Kod 17H wskazuje na TAB — ustawiamy odpowiednie wskazniki. Realizacja TAB następuje w liniach 430—500. Tam rozpoznaje się relację pomiędzy aktualną pozycją druku (POZYCE) i zawartością TABBUF. Dla POZYCE mniejszego od TABBUF dodrukowuje się wymaganą liczbę spacji.



Listing 1

```
[illegible] drukarki
[illegible]
[illegible]
[illegible] CHAN[illegible]      EQU   #5[illegible]
[illegible]
[illegible]                      ORG   5[illegible]
[illegible]
[illegible] ;Procedura inicjujaca
[illegible] ;kanal obslugi drukarki
100
110 INIT        LD    HL,CHAN[illegible]
120             LD    BC,15
130             ADD   HL,BC
140             LD    BC,MP[illegible]
150             LD    (HL),[illegible]
160             INC   HL
170             LD    (HL),B
180             LD    B,8
190             LD    HL,OUTBUF
200 ZERUJ       LD    (HL),0
210             INC   HL
220             DJNZ  ZERUJ
230             JP    UNLIS0
240
250 ;Transkoder Spectrum =
260 ; >= drukarka MPS-803
270 MPS803      LD    HL,STATFL
280             BIT   7,(HL)
290             SET   7,(HL)
300             CALL  Z,INICJ
310             CP    #0F
320             JP    Z,STER2
330             BIT   4,(HL)
340             JP    NZ,DRUK1
350             BIT   6,(HL)
360             JR    Z,WEJ
370             BIT   5,(HL)
380             RES   5,(HL)
390             JR    Z,TAB1
400             LD    (TABBUF),A
410             RET
420
430 TAB1        RES   6,(HL)
440 TAB2        LD    HL,(TABBUF)
450             LD    A,L
460             SUB   H
470             RET   Z
480             RET   [illegible]
490             CALL  SPCOUT
500             JR    TAB2
510
520 WEJ         CP    #20
530             JR    C,STER
540             CP    #41
550             JR    C,DRUK3
560             CP    #5B
570             JR    NC,INNYZN
580             ADD   A,#20
590 DRUK3       JP    DRUK
600
610 INNYZN      CALL  ROZPOZ
620             JR    C,DRUK3
630             CP    #7B
640             JR    NC,DALEJ
650             SUB   #20
660             JR    DRUK3
670
680 DALEJ       CP    #91
690             JR    Z,DRUK4
700             CP    #[illegible]
710             JR    Z,DRUK4
720             SUB   #A5
730             JR    C,SPCOUT
740 TOKEN       LD    DE,#95
750             PUSH  AF
760             CALL  #0C41
770             CALL  NC,SPCOUT
780 POWT        LD    A,(DE)
790             AND   #7F
800             CALL  WEJ
810             LD    A,(DE)
820             INC   DE
830             ADD   A,A
840             JR    NC,POWT
850             POP   [illegible]
860             CP    #48
870             JR    Z,TOKEN1
880             CP    #82
890             RET   C
900 TOKEN1      LD    A,D
910             CP    3
920             RET   C
930 SPCOUT      LD    A," "
940             JR    DRUK
950
960 STER        CP    #08
970             JR    NZ,INST1
980             SET   4,(HL)
990 DRUK5       LD    HL,TABBF2
1000            LD    (HL),0
1010 [illegible] JR   DRUK2
[illegible]
[illegible]     CP    #0F
[illegible]     JR    NZ,INST2
[illegible]     CALL  DRUK2
[illegible]     RES   4,(HL)
[illegible]     LD    [illegible],(TABBF2)
[illegible]     LD    HL,POZYCE
[illegible]     INC   [illegible]
[illegible]
[illegible]
[illegible]
1130            JR    [illegible]RPOZ
1140
1150            [illegible] #1[illegible]
1160            JR    NZ,[illegible]
1170            [illegible] [illegible],HL
1180            SET   [illegible],HL
1190            RET
1200
1210            CP    [illegible]
1220            JR    Z,KOMMA
1230            CP    #0[illegible]
1240            JR    NZ,DRUK2
1250            RES   7,(HL)
1260            DEC   HL
1270            LD    (HL),0
1280            CALL  DRUK5
1290            JP    UNL[illegible]T
1300
1310 KOMMA      CALL  SPCOUT
1320            LD    A,(POZYCE)
1330            AND   #0F
1340            JR    NZ,KOMMA
1350            RET
1360
1370 INICJ      PUSH  AF
1380            PUSH  HL
1390            LD    A,4
1400            CALL  CZYTAJ
1410            LD    A,103
1420            CALL  SEKALI
1430            POP   HL
1440            POP   AF
1450            RET
1460
1470 ROZPOZ     PUSH  HL
1480            LD    HL,TABZN-2
1490            LD    B,8
1500 SZUK       INC   HL
1510            INC   HL
1520            CP    (HL)
1530            JR    Z,ZNAL
1540            DJNZ  SZUK
1550            CP    #61
1560            POP   HL
1570            RET
1580 ZNAL       INC   HL
1590            LD    A,(HL)
1600            POP   HL
1610            SCF
1620            RET
1630
1640 TABZN      DEFB  #5C,[illegible]
1650            DEFB  #5F,#A4
1660            DEFB  #60,#5C
1670            DEFB  #7B," "
1680            DEFB  #7C,#7D
1690            DEFB  #7D," "
1700            DEFB  #7E,"[illegible]
1710            DEFB  #7F,"[illegible]
1720
1730 DRUK       LD    HL,POZYCE
1740            INC   (HL)
1750 DRUK2      JP    IECOUT
1760 DRUK1      LD    HL,TABBF2
1770            INC   (HL)
1780            JR    DRUK2
1790
1800 ;Handler magistrali IEC625
1810 CZYTAJ     OR    %00100000
1820 ROZKAZ     PUSH  AF
1830            LD    HL,OBUFFL
1840            BIT   7,(HL)
1850            JR    Z,CZYTA1
1860            RES   7,(HL)
1870            INC   HL
1880            SET   7,(HL)
1890            PUSH  HL
1900            CALL  BYTOUT
1910            POP   HL
1920            RES   7,(HL)
1930 CZYTA1     POP   AF
1940            LD    (OUTBUF),A
1950            DI
1960            CALL  KASDAN
1970            CALL  USTATN
1980 WYSLIJ     DI
1990            CALL  USTZEG
2000            CALL  KASDAN
2010            XOR   [illegible]
2020 CZEKAJ     DEC   [illegible]
2030            JR    NZ,CZEKAJ
2040 BYTOUT     DI
2050            CALL  KASDAN
2060            CALL  ODCZYT
2070            JR    NC,BLAD
2080            CALL  KASZEG
2090            LD    HL,EOIFLG
2100            BIT   7,(HL)
2110            JR    Z,BYTOU2
2120 BYTOU0     CALL  ODCZYT
2130            JR    C,BYTOU0
2140 BYTOU1     CALL  ODCZYT
2150            JR    NC,BYTOU1
2160 BYTO[illegible] CALL [illegible]YT
2170            JR    C,BYTO[illegible]
2180            CALL  [illegible]TZE[illegible]
2190            LD    B,[illegible]
2200 BYT[illegible] CALL [illegible]YT
2210            [illegible] [illegible],BLA[illegible]
2220            [illegible] [illegible]TBUF
[illegible]
[illegible]
[illegible]
2260            JR    BYOULH
2270
2280 BYOUTH     CALL  KASDAN
2290 BYOULH     CALL  [illegible]
2300            CALL  USTZEG
2310            CALL  KASDAN
2320            DJNZ  BYTOL[illegible]
2330            LD    B,#18
2340 BYTO[illegible] DEC B
2350            [illegible] Z,[illegible]
2360            CALL  [illegible]ZYT
2370            JR    NC,BY[illegible]U4
2380            EI
2390            RET
2400
2410 IE[illegible] PUSH HL
2420            LD    HL,OBUF[illegible]L
2430            BIT   7,(HL)
2440            SET   7,(HL)
2450            PUSH  BC
2460            PUSH  AF
2470            CALL  NZ,BY[illegible]UT
2480            POP   AF
2490            POP   BC
2500            LD    (OUTBUF),A
2510            POP   HL
2520            RET
2530
2540 SEKALI     LD    (OUTBUF),A
2550            CALL  WYSLIJ
2560 KASATN     LD    A,%00100000
2570            JR    KASUJ
2580
2590 UNLIST     LD    A,63
2600            CALL  ROZKAZ
2610 UNLIS0     CALL  KASATN
2620            CALL  KASZEG
2630            JR    KASDAN
2640
2650 BLAD       EI
2660            CALL  UNLIS0
2670            LD    HL,0
2680            LD    (STATFL),HL
2690
2700 ;meldunek
2710 ;Invalid Stream
2720
2730            RST   [illegible]
2740            DEFB  23
2750
2760 KASZEG     LD    A,%01000000
2770            JR    KASUJ
2780
2790 KASDAN     LD    A,%10000000
[illegible] KASUJ PUSH HL
2810            CPL
2820            LD    HL,WZOR
2830            AND   (HL)
2840            JR    USTAW1
2850
2860 USTATN     LD    A,%00100000
2870            JR    USTAW
2880
2890 USTZEG     LD    A,%01000000
[illegible]     JR    USTAW
2910
2920 USTDAN     LD    A,%10000000
2930 USTAW      PUSH  HL
2940            LD    HL,WZOR
2950            OR    (HL)
2960 USTAW1     OUT   (251),A
2970            LD    (HL),A
2980            POP   HL
2990            RET
3000
3010 ODCZYT     CALL  #1F54   ;Test
3020            JR    [illegible],BLAD ;BREAK
3030            IN    A,(251)
3040            AND   %11000000
3050            LD    C,A
3060            [illegible] A,(251)
3070            [illegible] %11000000
3080            CP    C
3090            JR    NZ,ODCZYT
3100            ADD   A,A
3110            RET
3120
3130
3140 OUTBUF     DEFB  0
3150 OBUFFL     DEFB  0
3160 EOIFLG     DEFB  [illegible]
3170 WZOR       DEFB  [illegible]
3180 TABBUF     DEFB  [illegible]
3190 POZYCE     DEFB  [illegible]
3200 STATFL     DEFB  [illegible]
3210 TABBF2     DEFB  [illegible]
3220
3230 *KONIEC PROGRAMU
```

Kod sterujący 6 rozpoznaje się w linii 1210 — jest to przecinek przy PRINT, czyli druk od najbliższej pozycji będącej wielokrotnością 16. Procedurę ustalania tej pozycji realizuje fragment ▪ linii 1310—1350; spacje są drukowane tak długo, aż na najmłodszych 4 bitach POZYCE będą same zera.

W linii 1230 sprawdzamy akumulator [illegible] kod ENTER. Dla kodów pozostałych nie stosujemy już żadnej procedury przekodowania — są one bezpośrednio wysyłane na drukarkę. Dla ENTER natomiast należy wyzerować POZYCE, TABBF2 (CALL DRUK5 — przy okazji wysyłamy kod znaku) oraz zakończyć transmisję (JP UNLIST) — [illegible] to w tym celu, aby po wydrukowaniu każdej nowej linii odblokować magistralę i uaktywnić funkcję LF drukarki. W tym miejscu trzeba powiedzieć, że operacja ta nieco spowalnia pracę drukarki, lecz umożliwia za to jej pełne wykorzystanie i wyłączanie na czas trwania przerwy w drukowaniu (nie trzeba na nowo inicjować handlera).

Linie od 1730 do 1780 realizują wysyłanie znaków do drukarki. Dla skoku do DRUK następuje przy tym zwiększenie pozycji druku o 1 (POZYCE), dla DRUK1 — pomocniczego bufora TABBF2, przy druku graficznym, [illegible] dla DRUK2 — nie zmieniamy wartości tych zmiennych (przy wysyłaniu znaków sterujących).

Program ładujemy od adresu 65000 (dla ORG, jak na listingu 1) po operacji CLEAR 64999 i inicjujemy jego pracę przez RANDOMIZE USR 65000. Po tej operacji działają funkcje LPRINT i LLIST, także przy współpracy [illegible] INTERFACE 1 (program handlera plottera VC1520 zamieszczony w „MT" 9/86 pracował w tym przypadku błędnie — wymagana jest zmiana fragmentu procedury

Listing 2

```
10
20 MPSOUT        EQU  31491;wejscie
30 ;handlera przy jego poczatku
40 ,od adresu 31465 (ORG)
[illegible]
60               ORG  31265
70
80 WEJ           CP   #22
90               JR   Z,CUDZ
[illegible]      CP   #20
110              JP   NC,MPSOUT
120              CP   6
130              JR   C,DRUKGR
140              CP   #18
150              JR   C,INZNAK
160              SUB  #0E
170              JR   DRUKGR
[illegible] CUDZ SUB  #10
190              JR   DRUKGR
200 INZNAK       CP   #0F
210              JR   Z,APOLM
220              CP   #0C
230              JP   NC,MPSOUT
240              SUB  3
250              JR   DRUKGR
260 APOLM        SUB  [illegible]
270 DRUKGR       AND  A
280              RLA
290              LD   D,A
[illegible]      RLA
310              RLA
320              SUB  D
330              LD   HL,TABLZN
340              ADD  A,L
350              LD   L,A
[illegible]      LD   A,H
370              ADC  A,#0
[illegible]      LD   H,A
[illegible]      LD   A,8
400              PUSH HL
410              CALL MPSOUT
420              [illegible] HL
430              LD   B,6
[illegible] WYPR LD   A,(HL
[illegible]      INC  HL
[illegible]      PUSH HL
[illegible]      PUSH [illegible]
[illegible]      CALL MPSOUT
[illegible]      POP  [illegible]
[illegible]      POP  HL
[illegible]      [illegible] NZ WYPR
[illegible]      [illegible] A #0F
[illegible]      [illegible] MPSOUT
[illegible]      DEFB #F[illegible] #[illegible] #E[illegible] #[illegible] #F[illegible]
[illegible]      DEFB #BE #C[illegible] #C[illegible] [illegible] #A[illegible]
[illegible]      DEFB #FF,#C9,#C9,#C9,#C1,#A0
570              DEFB #FF,#C8,#C4,#C0,#C0,#80
[illegible]      DEFB #FF,#84,#8A,#91,#FF,[illegible]
590              DEFB #BE,#C1,#C3,#C1,#BE,#80
600              DEFB #A6,#C9,#C9,#CB,#B1,#80
610              DEFB #E1,#D1,#CB,#C5,#C[illegible],#80
620              DEFB #E2,#D1,#C9,#C5,#C[illegible] [illegible]
630              DEFB #90,#AA,#AA #9E,#A0,#C0
640              DEFB #B8,#C4,#C6,#C5,#C4,#80
[illegible]      DEFB #9[illegible],#AA,#AA,#AA,#C4,#80
[illegible]      DEFB #80,#91,#FF,#C4,#80,#80
[illegible]      DEFB #84 #F8 #84 #8[illegible],#F9,#80
[illegible]      DEFB #B[illegible],#C4,#C6,#C[illegible],#B8,#80
[illegible]      DEFB #C[illegible],#D4,#D[illegible],#D[illegible],#A0,#80
[illegible]      DEFB #C4,#E4,#D[illegible],#C[illegible] #C4,#[illegible]
[illegible]      DEFB #C4,#E4,#D6,#[illegible],#C4,#8[illegible]
[illegible]      DEFB #80,#87,#E[illegible],#[illegible],#8[illegible] #[illegible]
[illegible]

[illegible] PROGRAMU*
```

inicjalizacji). Można oczywiście program ulokować w innym miejscu pamięci, pod warunkiem zmiany ORG. Należy przy tym pamiętać, że program liczy łącznie 535 bajtów.

Kilka słów na temat zmian w handlerze magistrali. Po pierwsze wprowadzono w miejsce zmiennej KANAFL pomocniczy bufor TABBF2 (linia 3210). Po drugie wprowadzono testowanie BREAK — linie 3010 i 3020 — dla BREAK następuje skok do obsługi błędu i meldunek („Invalid stream"). Po takim przerwaniu transmisji konieczne jest wyzerowanie drukarki (wyłączenie z sieci) i ponowna inicjalizacja handlera (RANDOMIZE USR 65000 lub inny adres).

Poważną wadą drukarek jest brak polskich liter. Jest to szczególnie widoczne przy korzystaniu z polskich wersji edytorów. Litery te można jednak stosunkowo łatwo wprowadzić wykorzystując tryb druku graficznego. W polskiej wersji TASWORD-a następujące kody oznaczają polskie litery:

Ą — ØC	ą — ØF
Ć — Ø1	ć — 18
Ę — [illegible]2	ę — 19
Ł — Ø3	ł — 1A
Ń — Ø4	ń — 1B
Ó — Ø5	ó — 1C
Ś — Ø7	[illegible] — 1D
Ź — ØA	[illegible] — 1E
Ż — [illegible]	ż — 1F

Kody te podano oczywiscie heksadecymalnie

Przy druku graficznym występuje jednak jeszcze jeden problem. Otóż po wydrukowaniu cudzysłowu (") drukarki MP8801 i MP8803 drukują wszystkie kody sterujące w postaci liter w rewersie — a zatem kod włączenia druku graficznego także. Należy zatem cudzysłów także wydrukować w trybie graficznym — pozwoli to na uniknięcie nienormalnej pracy drukarki.

Program drukujący graficznie polskie litery oraz cudzysłow przedstawia listing 2. W liniach od 80 do 260 następuje rozpoznanie kodow i w przypadku kodow polskich liter oraz cudzysłowu wprowadzenie odpowiedniej poprawki w akumulatorze [illegible] po tej operacji zawiera on ko[illegible] numer znaku [illegible] tabeli (dla innych kodow następuje skok do handlera dru-karki). Numer ten należy pomnożyć przez 6 (ilość kolumn na znak) — linie 270—320. W ten sposób otrzymujemy poprawkę, którą następnie dodajemy do adresu tablicy TABLZN. Teraz należy tylko włączyć tryb graficzny (linia 390—420), wysłać 6 bajtów do drukarki (linia 440—510) oraz wyłączyć tryb graficzny (linie 520 i 530).

Program ten, łącznie z handlerem drukarki, należy umieścić poniżej adresu 32000 — od niego zaczyna się bufor tekstu TASWORD-a. Aby program ten poprawnie współpracował [illegible] handlerem, należy wnieść do niego dwie poprawki (listing 3). Po pierwsze zmieniamy ORG, po drugie w tablicy adresów procedury wyjścia drukarki należy wpisać wejście programu przeko-

Listing [illegible]

```
 10 ;Handler drukarki
 20 ;MPS-801 i MPS-803
 30
 40 CHANS        EQU  #5C4F
 50
[illegible]      ORG  31465
 70
 80 ;Procedura inicjujaca
[illegible] kanal obslugi drukarki
100
110 INIT         LD   HL,(CHANS)
120              LD   BC,15
130              ADD  HL,BC
140              LD   BC,31265
150              LD   (HL),C
160              INC  HL
170              LD   (HL),B
180              LD   B,8
190              LD   HL,OUTBUF
[illegible] ZERUJ LD  (HL),0
210              INC  HL
220              DJNZ ZERUJ
230              JP   UNL130
```

dowującego (z listingu 2). Można oczywiscie wpisac pod asemblerem oba programy naraz, lecz nie zawsze jest to wygodne — można przeprowadzic dwukrotnie asemblację otrzymując w wyniku 735 bajtow kodu wynikowego.

Program wynikowy (rozpoczynający się od adresu 31265) inicjujemy od adresu 31465 — jest to adres procedury inicjalizacji handlera ze zmienionym w tym przypadku adresem wejscia. Inicjalizację tę należy przeprowadzić jednak po wykonaniu operacji na tablicy adresów kanałow, realizowanej w edytorze TASWORD po przez GOSUB 0. Konieczna jest też oczywiscie instrukcja CLEAR 31264, a także dość powazne skrócenie części napisanej w BASIC-u w celu wygospodarowania miejsca na handler. Najlepiej wykonac to likwidując zbędne napisy, względnie niepotrzebne funkcje (choćby chowania TASWORD-a na tasmę)

Celowo nie podajemy programu ładującego napisanego w BASIC-u. Byłby on wątpliwym ułatwieniem wprowadzania programu stwarzając nieco więcej okazji do popełnienia błedu. Ponadto naprawdę warto [illegible] posługiwać się asemblerem.

[illegible]

ZEGAR CYFROWY W ZX SPECTRUM

Roland Wacławek

Programowanie zegara cyfrowego jest często jednym z pierwszych ćwiczeń początkujących programistów. Łatwo napisać go w języku BASIC. Niestety, taki zegar absorbuje procesor bez reszty i uniemożliwia wykorzystanie go do innych, bardziej pożytecznych zadań. Praktyczniej byłoby mieć zegar „wmontowany"na stałe w kąt ekranu, wskazujący stale aktualny czas, ale nie przeszkadzający np. w programowaniu lub wprowadzaniu danych. Zegar ten musiałby być obsługiwany przez procesor „dorywczo", bez odciągania go od ważniejszych prac. Realizacja takiego zegara wymaga sięgnięcia po język asemblera i dostępny w ZX Spectrum mechanizm przerwań zegarowych.

Przerwania zegarowe polegają na tym, że 50 razy na sekundę ULA wysyła na końcówkę IRQ (wejście przerwań maskowalnych) procesora Z80 sygnał, żądający obsługi przerwania. Jeśli w tym momencie przerwania IRQ były dozwolone (jest tak po wykonaniu rozkazu EI), procesor zawiesza realizację aktualnego programu, zapamiętuje stan licznika rozkazów (jak po CALL) i skacze do tzw. podprogramu (procedury) obsługi przerwania. Po wykonaniu stosownych czynności (w przypadku Spectrum: odczyt klawiatury i wyznaczenie następnika czyli zwiększenie o 1 zmiennej systemowej FRAMES) procesor powraca do realizacji przerwanego programu (jak po RET). Procedura obsługi musi na wstępie przechować na stosie, a przed zakończeniem — odtworzyć zawartość wszystkich używanych przez nią rejestrów, łącznie z rejestrem stanu F. W chwili wznowienia realizacji zawieszonego programu procesor musi być bowiem w identycznym stanie, jak w chwili przerwania.

ZX Spectrum dysponuje, co prawda, prymitywnym zegarem wewnętrznym (wspomniana już, trzybajtowa zmienna FRAMES), ale do naszych celów jest on nieprzydatny. Skonstruujemy zatem nasz zegar cyfrowy od podstaw sami.

Na czym polega działanie zegara mechanicznego? Na zliczaniu impulsów, dostarczanych przez miarowo tykający balans. Nasz zegar cyfrowy będzie funkcjonował całkiem podobnie. Rolę balansu przejmie jednak mechanizm przerwań, a kilkustopniowy licznik skonstruujemy nie z kołek zębatych, lecz z liczb dwójkowych.

Program zegara cyfrowego (prog. 1) składa się z dwóch wyraźnie wyodrębnionych części. Pierwsza zlicza impulsy zegarowe (linie: 110—250), druga wyświetla na ekranie aktualny stan zegara (od etykiety KOPIUJ). Aktualny stan zegara przechowują cztery liczniki: oddzielnie dla godzin, minut, sekund i „tyknięć", czyli elementarnych impulsów, generowanych przez system przerwań w ZX Spectrum. Na jedną sekundę składa się 50 takich „tyknięć". Nasz program zegarowy będzie po prostu nową procedurą obsługi przerwań zegarowych. Spowodujemy, że po każdym przerwaniu program zegara będzie uruchamiany od nowa (od etykiety: ZEGAR).

Pierwszą czynnością programu jest zwiększenie o 1 licznika „tyknięć" i sprawdzanie, czy aby nie nastąpiło jego przepełnienie. Przez przepełnienie licznika rozumiemy sytuację, w której jego stan zrównał się z pojemnością, inaczej: zawartość licznika jest o 1 większa od najwyższej dopuszczalnej (licznik do 10 może przyjmować wartości 0, 1....8, 9).

W razie przepełnienia licznik zostaje wyzerowany (będzie zliczał od 0), a równocześnie następuje inkrementacja licznika następnego w kolejności, w naszym przypadku: licznika sekund (zostaje uwzględnione tzw. przeniesienie). Kiedy i ten licznik ulegnie przepełnieniu, nadejdzie kolej powiększenia licznika minut, itd. Tylko w razie przepełnienia ostatniego licznika — godzin — przeniesienie nie będzie brane pod uwagę. Zgrupowane w tablicy CZAS liczniki mają różne pojemności, dlatego pojemność każdego z nich jest odczytywana z odrębnej komórki tablicy MAKSYM.

Po zarejestrowaniu „tyknięcia" i odpowiednim zaktualizowaniu stanu liczników program przystępuje do wyświetlania stanu zegara. Na ekranie widoczne są tylko sekundy, minuty i godziny. Chociaż wskazania zmieniają się co sekundę, obraz zegara jest odświeżany co 20 ms. Dzięki temu pole czasu nie znika nawet przy częstym scrollingu i ciągłej zmianie treści ekranu.

Wyświetlanie sześciu cyfr wydaje się proste — odpowiednie narzędzia są przecież dostępne w pamięci ROM. Używa ich m.in. instrukcja PRINT. Niestety, korzystanie z nich w programie obsługi przerwania byłoby nader kłopotliwe. Dlaczego? Otóż procedury wyprowadzania znaków w ROM korzystają z wielu zmiennych systemowych, zapisanych w pamięci ROM. Tymczasem przerwania zegarowe występują także podczas realizacji programu podstawowego, napisanego w języku BASIC lub innym. Przypuśćmy, że przerwanie zegarowe nastąpiło w trakcie wykonywania instrukcji PRINT. Przy wyświetlaniu wskazań zegara zostanie zmieniona pozycja kursora, wartości atrybutów itd. Po zakończeniu obsługi przerwania, realizacja instrukcji PRINT będzie kontynuowana. Ponieważ jednak w międzyczasie w „podstępny" sposób zostały zmienione ustawione już parametry, efekty na ekranie mogą być, delikatnie mówiąc, dziwne. Chcąc korzystać z procedur w ROM należałoby zatem przechowywać na stosie nie tylko zawartości rejestrów procesora, ale i wszystkie zmienne systemowe, które są modyfikowane przy wyprowadzaniu informacji na ekran. Lepiej już stworzyć własne procedury obsługi ekranu — będzie to zarazem niezła wprawka dla adeptów asemblera.

Do wyprowadzania na ekran pojedynczego znaku służy procedura WYPROW. W chwili jej wywołania akumulator musi zawierać kod znaku, a rejestr L — numer pola ekranu, w którym znak ma być umieszczony. W grę wchodzi górna tercja. Pola 0...31 leżą w pierwszej linii, 32..63 w drugiej, itd. Procedura WYPROW wykorzystuje fakt, że adresy: bajtu atrybutów i wszystkich ośmiu składających się na reprezentację znaku bajtów pamięci ekranu mają identyczny młodszy bajt, zaś starszy

bajt wskaźnika pamięci ekranu przy posuwaniu się od linii do linii wzrasta o 1.

WYPROW najpierw kompletuje adres bajtów atrybutów i wpisuje jego nową wartość: białe tło, czarny atrament, podwyższona jaskrawość (można sobie zdefiniować inne). Następnie WYPROW tworzy w HL adres bajtu pamięci ekranu, opisującego wygląd górnych ośmiu punktów znaku (górnego wiersza pola). Pomnożywszy kod znaku przez 8 uzyskujemy tzw. przesunięcie (ang. *offset*) względem bazy tablicy wzorców znaków (wzorce są ułożone według wartości kodów, każdy z nich zajmuje 8 bajtów). Mnożenie przez 8 najprościej zrealizować, dodając zawartość rejestru trzykrotnie do siebie. Każde dodawanie podwaja jego zawartość, 2↑3 = 8. Po zsumowaniu przesunięcia z adresem bazowym, zawartym w zmiennej systemowej CHARS, uzyskujemy efektywny adres wzorca. Pozostaje przepisać osiem kolejnych bajtów wzorca do komórek pamięci ekranu, odpowiadających ośmiu położonym jeden pod drugim wierszom znaku. Odbywa się to w pętli. Po każdym powtórzeniu wskaźnik pamięci ekranu wzrasta o 256 (INC H), zaś wskaźnik bajtów wzorca — o 1 (INC E). Zamiast INC E można użyć bardziej ogólnego rozkazu INC DE. Ponieważ wzorce znaków są ułożone w ROM w taki sposób, że żaden z nich nie przekracza granicy strony, rozkaz INC E spełni podobną funkcję. Na zakończenie procedura WYPROW wyznacza jeszcze następnik (inkrementuje) zawartość L, która dzięki temu wskazuje następne w kolejności pole ekranu.

Procedura WYPROW, choć może działać samodzielnie, stanowi integralny fragment procedury DWIECY, wyprowadzającej jednobajtową liczbę z zakresu 0...99 jako ciąg dwóch cyfr dziesiętnych. Zamiana z postaci dwójkowej na dziesiętną odbywa się metodą dzielenia całkowitego (dzielenia z resztą) wyprowadzanej liczby przez współczynniki wagowe kolejnych cyfr, poczynając od najstarszej. Po zakończeniu każdego etapu iloraz przedstawia wartość cyfry, zaś reszta z dzielenia zostanie dzielną w następnym kroku. W naszym przypadku są tylko dwa kroki: dzielenie przez 10 i przez 1. Dzielenie odbywa się drogą cyklicznego odejmowania dzielnika, dopóki różnica jest nieujemna. Liczba odejmowań jest ilorazem. Ponieważ interesuje nas nie tyle iloraz, co efektywny kod cyfry, licznik cykli (rejestr C) jest ładowany początkowo kodem cyfry „0" pomniejszonym o 1 (poprawka na ostatni, unieważniany cykl, w którym różnica zmienia znak). Dzięki temu po zakończeniu pętli C zawiera wprost kod znaku. Ponieważ w ostatnim odejmowaniu wystąpił wynik ujemny, dla odtworzenia poprawnej reszty należy do zawartości akumulatora dodać odjemnik. Anuluje to skutki ostatniego rozkazu SUB B.

Zastosowane procedury wyświetlania mają w stosunku do tych z ROM ważną zaletę: działają szybciej. Dzięki temu spowolnienie pracy Spectrum, związane z „zakulisową" pracą zegara, jest pomijalne. Po wyprowadzeniu na ekran aktualnego czasu program oddaje sterowanie standardowej procedurze obsługi przerwań, umieszczonej w ROM. Jest to konieczne, gdyż procedura ta zapewnia m.in. poprawną pracę klawiatury.

Aby zegar „tykał", należy spowodować, aby po każdym przerwaniu wywoływana była nie standardowa procedura obsługi z ROM, lecz nasz program. Z80 w ZX Spectrum i TIMEX 2048 normalnie obsługuje przerwania maskowalne (IRQ) w trybie 1. Oznacza to, że każdy sygnał przerwania IRQ uruchamia podprogram pod adresem 56/$38 w pamięci ROM. Aby zmienić ten adres, trzeba przełączyć procesor w tryb 2, tzw. wektoryzowany. W trybie tym wyznaczanie adresu procedury obsługi odbywa się dwuetapowo. Najpierw Z80 odczytuje bajt z magistrali, który normalnie powinien być tzw. wektorem przerwania, wysłanym przez to urządzenie zewnętrzne, które przerwanie spowodowało. Bajt ten wspólnie z zawartością rejestru I tworzy pełny adres słowa w pamięci operacyjnej, zawierającego właściwy (efektywny) adres procedury obsługi. Rejestr I, dostarczający ośmiu starszych bitów adresu pośredniego, zawiera numer tzw. stronicy PAO (bloku 256 bajtów, których adresy różnią się tylko w młodszym bajcie), na której mieści się tablica procedur obsługi.

Prog. 1 Postać źrodłowa programu maszynowego ZEGAR-MT
Użyto asemblera HISOFT GENS 3

```
        10 ;ZEGAR-MT dla ZX Spectrum/TIMEX 2048. R. Waclawek [illegible]
        20
0078    30 ATRYB  EQU  %01111000   ;bajt atrybutow  cyfr zegara
0058    40 HATRYB EQU  88          ;st. bajt adresu pam. atryb.
0040    50 HEKRAN EQU  64          ;st. bajt adresu pam. ekranu
5C36    60 CHARS  EQU  #5C36       ;adres adresu generatora zn.
00FE    70 PWEKT  EQU  #FE         ;st. bajt adr. pola wektorow
00FC    80 ADRBYT EQU  #FC         ;ml. i st. bajt adr. przerw.
        90
FCFC   100        ORG  #FCFC
FCFC   110 ZEGAR  PUSH AF          ;tutaj  skok  po  przerwaniu
FCFD   120        PUSH HL          ;przechowaj rejestry uzywane
FCFE   130        PUSH DE          ;przez progr. obslugi zegara
FCFF   140        PUSH BC
FD00   150        LD   B,4         ;B:= ogolna liczba licznikow
FD02   160        LD   HL,CZAS+3   ;HL:=adres  licznika  1/50s.
FD05   170        LD   DE,MAKSYM+3 ;DE:=adr. granicy dla 1/50s.
FD08   180 NASTPO LD   A,(DE)      ;A:=wart. graniczna  pozycji
FD09   190        INC  (HL)        ;inkrementuj  stan  licznika
FD0A   200        CP   (HL)        ;czy przepelnienie licznika?
FD0B   210        JR   NZ,KOPIUJ   ;nie: wyswietl aktualny czas
FD0D   220        LD   (HL),0      ;wyzeruj stan  tego licznika
FD0F   230        DEC  HL          ;DE:=adres nastepn. licznika
FD10   240        DEC  DE          ;DE:=adres nastepnej granicy
FD11   250        DJNZ NASTPO      ;jesli nie ostatni, przenies
FD13   260 KOPIUJ LD   DE,CZAS     ;DE:= adres  licznika godzin
FD16   270        LD   L,24        ;L:=nr pola cyfry dz. godzin
FD18   280        CALL DWIECY      ;wyprowadz liczbe dwucyfrowa
FD1B   290        CALL DWUKDC      ;wypr. ":" i liczbe dwucyfr.
FD1E   300        CALL DWUKDC      ;wypr. ":" i liczbe dwucyfr.
FD21   310        POP  BC          ;odtworz rejestry BC oraz DE
FD22   320        POP  DE
FD23   330        JP   #38         ;skocz do proc. obsl. w ROM
       340
FD26   350 DWUKDC LD   A,":"       ;umiesc  w  A  kod dwukropka
FD28   360        CALL WYPROW      ;wyprowadz znak z A na ekran
FD2B   370 DWIECY LD   A,(DE)      ;A:=wartosc kolejn. licznika
FD2C   380        INC  DE          ;DE:=adres nastepn. licznika
FD2D   390        LD   B,10        ;B:=wsp. wagi dla dziesiatek
FD2F   400        CALL CYFRA       ;wyprowadz cyfre  dziesiatek
FD32   410        LD   A,B         ;reszta z dzielenia przez 10
FD33   420        LD   B,1         ;B:=wsp. wagi dla  jednostek
FD35   430 CYFRA  LD   C,"0"-1     ;C:=kod cyfry 0 pomniej. o 1
FD37   440 CYFRAP INC  C           ;ustaw w C kod kolejn. cyfry
FD38   450        SUB  B           ;odejmij kol. raz wage cyfry
FD39   460        JR   NC,CYFRAP   ;powtorz, jesli wynik nieuj.
FD3B   470        ADD  A,B         ;odtworz A sprzed ost.odejm.
FD3C   480        LD   B,A         ;B:=  reszta   z   dzielenia
FD3D   490        LD   A,C         ;ostateczny  kod  cyfry do A
FD3E   500 WYPROW PUSH BC          ;przechowaj rejestry BC i DE
FD3F   510        PUSH DE
FD40   520        LD   H, HATRYB   ;H:= st. bajt adr. pam. atr.
FD42   530        LD   (HL),ATRYB  ;zapisz bajt atrybutow znaku
FD44   540        LD   H, HEKRAN   ;H:= st. bajt adr. pam. ekr.
FD46   550        LD   E, A        ;kod znaku zaladuj do rej. E
FD47   560        LD   D, 0        ;DE:=kod wyswietlanego znaku
FD49   570        EX   DE,HL       ;przenies  kod  znaku  do HL
FD4A   580        ADD  HL,HL       ;podwoj zawartosc   pary  HL
FD4B   590        ADD  HL,HL       ;trzy razy pod rzad, mnozac
FD4C   600        ADD  HL,HL       ;kod  znaku  przez  liczbe 8
FD4D   610        LD   BC,(CHARS)  ;BC:=adres bazy gen.  znakow
FD51   620        ADD  HL,BC       ;wyznacz w HL  adres  wzorca
FD52   630        EX   DE,HL       ;adres  wzorca  znaku  do DE
FD53   640        LD   B, 8        ;B := licznik  linii   znaku
FD55   650 WYPROP LD   A, (DE)     ;laduj kol. bajt wzorca do A
```

```
FD56          LD   (HL),A      ;przenies bajt wzorca do RAM
FD57          INC  H           ;HL:= adr. nast. linii znaku
FD58          INC  E           ;DE:= adr. bast bajtu [illegible]
FD59          DJNZ WYPROP      ;powtorz,jesli nie osmy bajt
FD5B          POP  DE          ;odtworz rejestry DE oraz BC
FD5C          POP  BC
FD5D          INC  L           ;HL:=adres nast. znaku w RAM
FD5E          RET ;powrot z podprogramu
FD5F     CZAS DEFB 0           ;licznik  godzin
FD60          DEFB 0           ;licznik   minut
FD61          DEFB 0           ;licznik  sekund
FD62          DEFB 0           ;licznik 1/50 s.
FD63   MAKSYM DEFB 24          ;przepelnienie   dla  godzin
FD64          DEFB 60          ;przepelnienie   dla   minut
FD65          DEFB 60          ;przepelnienie   dla  sekund
FD66          DEFB 50          ;przepelnienie   dla 1/50 s.
```

ZX Spectrum i TIMEX 2048 nie są przystosowane do obsługi przerwań zegarowych w trybie 2, gdyż ULA nie wysyła wektora przerwania. Procesor odczyta z magistrali „w dobrej wierze" bajt, którego wartości w ogólnym przypadku nie sposób przewidzieć (zwłaszcza po dołączeniu do złącza krawędziowego dodatkowych interfejsów). Wskutek tego przy wyznaczaniu adresu pośredniego Z80 może zaadresować dowolną spośród 256 komórek stronicy pamięci, wyznaczonej zawartością rejestru I.

Zawartość tej stronicy musi być taka, aby odczytany adres efektywny nie zależał od adresu pośredniego. Stanie się tak wtedy, gdy wszystkie bajty tej strony oraz pierwszy bajt strony następnej otrzymają tą samą wartosc. Sprawa tego bajtu wymaga wyjaśnienia. Otóż w praktyce Z80 akceptuje także nieparzyste wektory przerwań. Po odczytaniu wektora o wartości 255 procesor odczyta młodszy bajt adresu efektywnego z ostatniej komórki stronicy wskazanej przez I, a starszy bajt — z następnej komórki, należącej już do stronicy następnej.

Korzystając w Spectrum z trybu 2 nie mamy więc pełnej swobody przy lokalizacji procedury obsługi przerwania: musimy umieścić ją pod adresem, spełniającym warunek: Adr = 257 * I, I: liczba całkowita 0..255. Liczby takie mają taki sam starszy i młodszy bajt, np.#A1A1 lub #FCFC. Przed przełączeniem procesora w tryb 2 trzeba jeszcze przygotować tablicę adresów, a numer stronicy, na której się znajduje, wpisać do rejestru I. Realizuje to procedura INICJZ (prog. 2). Wyłączenie zegara polega na przywróceniu trybu 1 (procedura KASUJZ). Oto adresy obu procedur, przydatne przy ich uruchamianiu z poziomu języka BASIC, np. przez RANDOMIZE USR

Adres: INICJZ = 64872; KASUJZ = 64893.

Prog. 2. Procedury uaktywniające i wyłączające program ZEGAR-MT

```
FD67   INICJZ LD   H,PWEKT     ;HL:=adres stronicy wektorow
FD69          LD   L,0
FD6B          LD   B,L         ;B bedzie licznikiem (0=256)
FD6C   INICJP LD   (HL),ADRBYT ;wypelniaj bajty  str. wekt.
FD6E          INC  HL          ;ustaw adres kolejnego bajtu
FD6F          DJNZ INICJP      ;powtarzaj, az zapiszesz 256
FD71          LD   (HL),ADRBYT ;zapisz  1 bajt nast. strony
FD73          LD   A,PWEKT     ;A:= wskaznik  pola wektorow
FD75          LD   I,A         ;zapisz  go  do  rejestru  I
FD77          IM   2           ;aktywizuj tryb przerwan IM2
FD79          RET
FD7A   KASUJZ IM   1           ;przywroc  tryb przerwan IM1
FD7C          RET
```

Prostą metodą nastawienia zegara jest wpisywanie instrukcją POKE stanu godzin, minut i sekund do odpowiednich komórek pamięci CZAS (adresy: 64863..64865). Modyfikując zawartość komórki nr 64807 mozemy zmienić separator pól (np. POKE 64807, CODE „—"). W komórce 64835 jest zapisany wzorzec bajtu atrybutów dla cyfr zegara, który też można dostosować do własnego gustu (np. POKE 64835, BIN 01000111 daje białe cyfry w czarnym polu). Wreszcie wpisując nową zawartość 0..247 do komórki nr 64791 jesteśmy w stanie przenieść „tarczę" zegara w dowolne miejsce w obrębie górnej tercji ekranu.

Aby zapewnic większy komfort obsługi zegara, program ZEGAR-MT został uzupełniony procedurą maszynową, umożliwiającą wygodne zatrzymywanie, uruchamianie i nastawianie zegara w języku BASIC. Pojedyncza instrukcja PRINT USR 64893, gg, mm, ss nastawia i równocześnie włącza zegar. gg, mm, ss to dowolne wyrażenia liczbowe, określające stan godzin, minut i sekund; w roli separatora musi wystąpić przecinek. PRINT USR 64893 bez dalszych argumentów wyłącza zegar. Listing (prog. 3) nie zawiera komentarzy.

Prog. 3 Procedura OBSZEG, pozwalająca nastawiać, włączac i wyłączac zegar przy pomocy instrukcji PRINT USR w języku BASIC

```
FD7D   OBSZEG RST  #18
FD7E          CALL #2045
FD81          JR   Z,WYLZEG
FD83          CALL #1C7D
FD86          CP   ","
FD88          JP   NZ,#1C8A
FD8B          CALL #1C79
FD8E          CALL #2307
FD91          LD   HL,CZAS+3
FD94          LD   (HL),0
FD96          DEC  HL
FD97          LD   (HL),B
FD98          DEC  HL
FD99          LD   (HL),C
FD9A          DEC  HL
FD9B          PUSH HL
FD9C          CALL #2314
FD9F          POP  HL
FDA0          LD   (HL),A
FDA1          CALL INICJZ
FDA4   KONIEC LD   SP,(#5C3D)
FDA8          JP   #1B76
FDAB   WYLZEG CALL KASUJZ
FDAE          JR   KONIEC
```

Rozszerzenia języka BASIC w ZX Spectrum i przekazywanie argumentow do podprogramów będą bowiem tematem odrębnego artykułu, który niebawem ukaże się na łamach „InforMika".

Śledząc na ekranie pracę zegara, można łatwo zaobserwować, w jakich sytuacjach obsługa przerwań zegarowych jest zawieszana w wyniku wykonania rozkazu maszynowego DI — np. w trakcie emisji dźwięku (BEEP) lub podczas zapisu lub odczytu z taśmy. W okresach tych zegar się zatrzymuje.

Cały pakiet procedur maszynowych zajmuje 180 bajtów i pierwotnie został zlokalizowany w komórkach 64764-64943. Do tego należy jeszcze doliczyć obszar 257 bajtów (adresy: 65024-65280) zarezerwowany dla tablicy adresów. Zarówno procedury, jak i tablicę adresów można umieścić w innym obszarze pamięci, przestrze-

gając wszakże podanych reguł. Jeśli do Spectrum nie są dołączone żadne przystawki, Z80 odczytuje standardową wartość wektora 255. W tym przypadku można zrezygnować z początkowych 255 bajtów tej tablicy. W przypadku dołączenia do złącza krawędziowego drukarek, interfejsow itd. program ZEGAR-MT może jednak nie działać.

Aby umożliwić szybkie wypróbowanie programu ZEGAR-MT i udostępnic go także tym Czytelnikom, którzy jeszcze nie są dość biegli w posługiwaniu się asemblerem, opracowano loader w języku BASIC (prog. 4).

Prog. 4 Program języku BASIC ładujący kod maszynowy do pamięci.

```
 10 REM LOADER programu maszynowego  ZEGAR-MI
 20 REM Roland Waclawek,Siemianowice Sl. 1986
 30
 40 CLEAR 64763: LET S= 0
 50 FOR A= 64764 TO 64943
 60    READ BAJT: POKE A, BAJT: LET S= S+BAJT
 70 NEXT A
 80 IF S<>19533  THEN PRINT "ZLE DANE!": STOP
 90
100 DATA 245,229,213,197,6,4,33,98,253,17,102
110 DATA 253,26,52,190,32,6,54,0,43,27,16,245
120 DATA 17, 95,253, 46, 24,205,43,253,205,38
130 DATA 253,205, 38,253,193,209,195,58, 0,62
140 DATA 58,205,62,253,26,19, 6,10,205,53,253
150 DATA 120, 6, 1,14,47,12,144,48,252,128,71
160 DATA 121,197,213,38,88,54,120,38,64,95,22
170 DATA 0,235,41, 41, 41,237,75,54,92, 9,235
180 DATA 6, 8,26,119,36,28, 16,250,209,193,44
190 DATA 201, 0,0,0,0,24,60,60,50,38,254,46,0
200 DATA 69, 54,252,35, 16,251, 54,252,62,254
210 DATA 237,71,237,94,201,237,86,201,223,205
220 DATA 69, 32, 40, 40,205,125,28,254,44,194
230 DATA 138,28,205,121,28,205,7,35,33,98,253
240 DATA 54, 0,43,112,43,113,43,229,205,20,35
250 DATA 225,119,205, 103,253, 237,123, 61,92
260 DATA 195,118,27,205,122,253,24,244
270
280 PRINT USR 64893,  19,30,00
```

Błędy powstałe przy wprowadzaniu zbioru DATA są sygnalizowane komunikatem. Po pomyślnym załadowaniu programu maszynowego do pamięci zegar jest niezwłocznie uruchamiany. Niech program ZEGAR-MT będzie inspiracją do samodzielnych prob „zakulisowego" zatrudnienia procesora z wykorzystaniem systemu przerwan zegarowych.

UWAGA KONKURS!

Redakcja „Młody Technik-InforMik" ogłasza konkurs na program edukacyjny, przygotowany na komputer pracujący w systemie MSX (najlepiej Spectravideo, sprzedawany w sklepach CSH).

Opracowane przez siebie programy prosimy nadsyłać pod adresem redakcji do 1987.12.31 na dyskietce lub kasecie (**gwarantowany zwrot!**)

Nagrodą w tym konkursie jest **100 szt dyskietek 5 1/4"** ufundowanych przez firmę PROSYSTEM z Wiednia oraz inne cenne nagrody.

NASZ TEST.

Timex Printer 2040

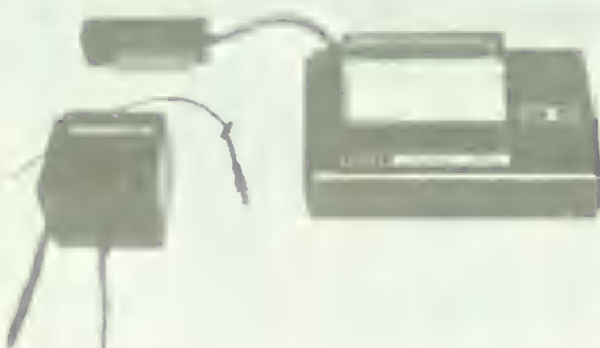
Jest to najprostsza drukarka współpracująca z mikrokomputerami typu ZX Spectrum — Timex. Druk odbywa się metodą termiczną na taśmie papieru termoczułego o szerokości 110 mm (4,33") rolce o maksymalnej średnicy 48 mm (1,9") i długości około 25 m (82 stopy). Uproszczona została do maksimum obsługa: drukarka posiada zaledwie 2 klawisze — włączenie zasilania z przesuwem taśmy oraz wyłączenie zasilania. Przy równoczesnym naciśnięciu obu klawiszy uzyskujemy wydruk testu drukarki: linie zapełnione zmianę cyframi 1 i 8. Drukarka dołączona jest do mikrokomputera poprzez złącze krawędziowe — niestety, w mikrokomputerze ZX Spectrum kabel ze złącza skutecznie utrudnia włożenie styku zasilania, co przy ogólnie przyjętym sposobie kasowania zawartości pamięci jest dosyć kłopotliwe. Karygodnym błędem producenta jest natomiast zastosowanie identycznych wtyków i gniazd zasilania w mikrokomputerze i drukarce. Nie byłoby problemu przy podobnych zasilaczach: mikrokomputer korzysta jednak zasilacza 9V 1,4 A, natomiast drukarka 24V 1,2 A! Nie trzeba wielkiej znajomości elektroniki, aby przewidzieć skutki niewłaściwego podłączenia zasilacza szczególnie do mikrokomputera z nie zabezpieczonymi elementami pamięci 4116 w przypadku awarii przetwornicy (przy tym zasilaniu bardzo prawdopodobnej). Niezbyt precyzyjnie wykonane gniazdo zasilania w drukarce dopuszcza także możliwość zwarcia na zasilaniu, co kończy się w najlepszym wypadku nadtopieniem mało odpornego na temperaturę tworzywa, z którego została wykonana obudowa gniazda. Należy tu nadmienić, iż momencie włożenia wtyczki zasilania do drukarki zaczynają pracować stabilizatory, czemu towarzyszy znaczny impulsowy pobór prądu.

Drukarka może być obsługiwana dodatkowymi instrukcjami Spectrum BASIC (LLIST, LPRINT, COPY) pozwalającymi wykonywać listowanie i pisanie oraz kopiowanie ekranu wprost na drukarce. Format wydruku jest identyczny jak organizacja ekranu w Spectrum, szybkość pisania ok. 1 linii/s tzn. około 30 zn/s. Jakość druku, raczej „wygrzewania" na papierze jest bardzo dobra, przy idealnie białej warstwie termoczułej druk jest bardzo kontrastowy (nadaje się szczególnie do listingów programów). Czy jednak CSH zapewniła niezbędne dostawy odpowiedniego papieru do tego typu drukarki? Bez możliwości zakupu papieru drukarka staje się jedynie kosztowną zabawką.

Ze względu sposób druku i zachodzące w drukarce szybkie zmiany stosunkowo dużych wartości prądów mogą wystąpić okresowe zakłócenia w odbiorze programów telewizyjnych (tzw. TVI). Należy wtedy wypróbować wszelkie możliwe ustawienia drukarki względem anteny telewizyjnej (i odwrotnie) lub po prostu nie drukować w czasie oglądania programu telewizyjnego. Szczególnej uwagi wymaga przechowywanie wydruków ze względu na specyficzne własności warstwy termoczułej. Konieczność unikania kontaktu z wysoką temperaturą jest oczywista, ponadto warstwa termoczuła jest wrażliwa na większe stężenia rozpuszczalników organicznych w powietrzu powodujących zaczernienie całej powierzchni wydruku. Na szczęście takich warunków nie spotykamy na co dzień.

Ogólnie należy ocenić Timex Printer 2040 jako udany wyrób przy postawionym zadaniu z zastrzeżeniem możliwości wykorzystania go w warunkach polskich wobec braku krajowej taśmy termoczułej do tego typu urządzenia.

Tadeusz Rzepecki



NASZ TEST:

URZĄDZENIE KASETOWE DATA RECORDER MK433

Pod koniec października ub. r. otrzymaliśmy dzięki uprzejmości Dyrekcji Zakładów Wytwórczych Magnetofonów w Lubartowie urządzenie kasetowe MK433 do przeprowadzenia prób eksploatacyjnych. Do chwili oddania niniejszego numeru do druku urządzenie to przepracowało ponad cztery miesiące przy współpracy z mikrokomputerem ZX Spectrum pracującym w systemie do edycji tekstów. W tym czasie dokonano zapisu ponad 10 kaset magnetofonowych (jednostronnie, kasety C60), co odpowiada ok. 2 MBajtom zapisanych danych. Urządzenie było eksploatowane dość intensywnie, lecz przy tym delikatnie, odpowiednio do wytrzymałości plastikowego mechanizmu. W czasie eksploatacji zaobserwowano następujące fakty mające wpływ na przydatność urządzenia:

1. W pierwszym okresie użytkowania następowało dość częste „wciąganie" taśmy, szczególnie na jej początku względnie przy kasetach o nieco większym tarciu krążków z taśmą. Przyczyną, która w pewnym momencie uniemożliwiła dalsze korzystanie z urządzenia, była wada fabryczna sprzęgła napędu wałka zdawczego taśmy. Po jej usunięciu wciąganie nie powtórzyło się.

2. W niektórych sytuacjach następowało opóźnione włączanie przewijania wstecz – konieczne było dwukrotne przesunięcie dźwigni przewijania.

3. Przy odczycie słabo nagranych taśm (z innego magnetofonu) występowały kłopoty, nawet mimo wystarczającego poziomu sygnału. Po odsunięciu urządzenia od monitora następowała pewna poprawa.

4. Przy stosowaniu taśm nowych, względnie o nieuszkodzonej warstwie magnetycznej, nie zaobserwowano żadnego przypadku przekłamania (błędu odczytu) w czasie całego okresu eksploatacji.

5. Próby stosowania programu Quick Save dawały poprawne efekty do ok. 4–5 stopnia. Powyżej 5 stopnia występował znaczny procent błędnych odczytów, niezależnie od typu zastosowanej taśmy.

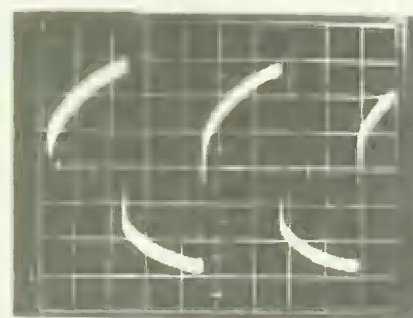
Subiektywna ocena pracy urządzenia MK433 daje w zasadzie (z zaniedbaniem usuniętej w pierwszym okresie wady sprzęgła) wynik bardzo dobry — wynika on z praktycznie zerowej stopy błędów. Należy jednak zaznaczyć, że na wynik ten ma znaczny wpływ kultura obchodzenia się z magnetofonem, w szczególności delikatne włączanie poszczególnych funkcji, jak również zachowanie odpowiedniej czystości i stosowanie właściwych taśm. Aby jednak dokonać także obiektywnej oceny parametrów, wykonano po pierwszych kilkunastu godzinach oraz na zakończenie testowania pomiary podstawowych parametrów:

- szybkości przesuwu taśmy
- charakterystyki zapis-odczyt, a konkretnie wierności zachowania kształtu impulsów.

Parametry te są najistotniejsze z punktu widzenia poprawności odczytu, gdyż mikrokomputer reaguje na pojawienie się zbocza impulsu (o dowolnym znaku) w określonym przedziale czasowym. Wyniki pomiarów są następujące:

- stałość szybkości przesuwu taśmy — lepsza od 2% przy temperaturze pokojowej, tj. 20±2 C
- wierność zachowania kształtu impulsów — przedstawiona na oscylogramach.

Widoczne jest bardzo dobre zachowanie kształtu impulsu, nie ma też charakterystycznego dla niektórych magnetofonów wahania amplitudy sygnału, wynikającego z niedokładności przylegania taśmy do głowicy. Zauważa się jednak dość znaczne pogrubienie przebiegu — wynika to z nałożenia nań sygnału pokazanego na fot. 5 – jest to przydźwięk pochodzący z układu odchylania poziomego monitora stojącego

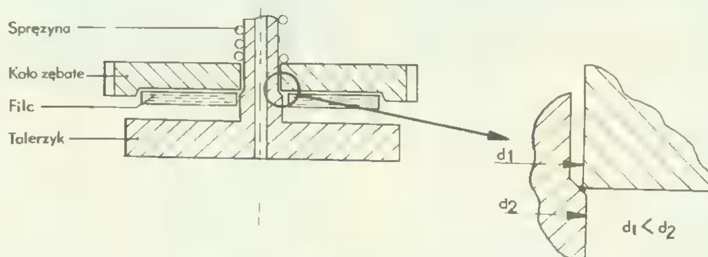


Fot. 1. — Oscylogram sygnału wyjściowego z gniazda MIC przy nagrywaniu ciągu zer, Y — 150 mV/dz, X — 200 μs/dz

w odległości ok. 30 cm od MK433. Nie występuje on oczywiście dla stłumionych wysokich tonów, lecz przy innych nastawach regulatora barwy dźwięku może wprowadzać trudności w odczycie słabszych nagrań — konieczne jest wtedy odsunięcie magnetofonu od monitora lub telewizora.

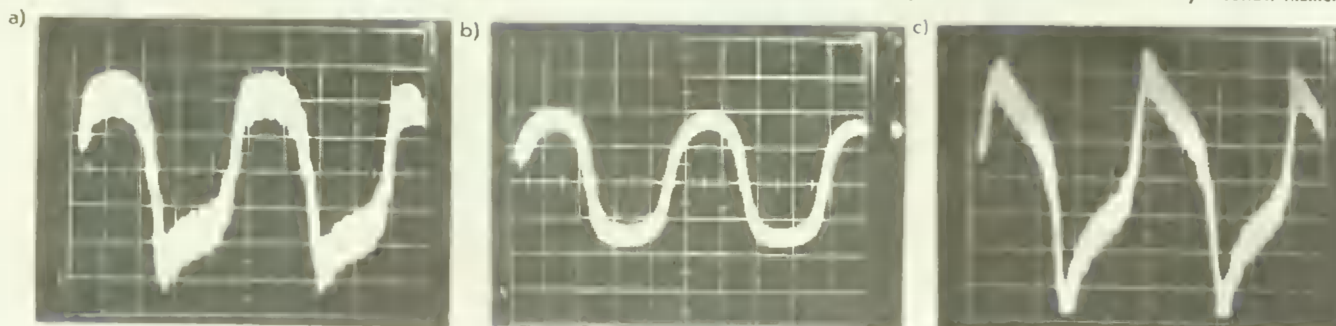
Pomiary przeprowadzone po zakończeniu okresu testowania nie wykazały zauważalnych różnic kształtu impulsu w procesie zapis-odczyt.

Reasumując — urządzenie podczas testu spisywało się bardzo dobrze, oczywiście po usunięciu wady sprzęgła, pokazanej schematycznie na rys. 1 – po rozwierceniu otworu w talerzyku do odpowiedniego wymiaru kłopoty z wciąganiem taśmy ustały. Wada ta wynika podobno z różnych wartości skurczu termicznego tworzyw stosowanych jako surowiec do wtryskowej produkcji detali napędu — należy o tym pamiętać, gdyż podobne wady mogą się powtarzać, a ich wykrycie jest czasem trudne (przy lekko obracających się taśmach wciąganie nie występowało). Należy przypuszczać, że tak dobre wyniki testu zwią-



Rys 1 — Wada sprzęgła: koło zębate nie jest dociskane do talerzyka przez krążek filcowy. Dla poprawnej pracy wymagane jest $d_1 < d_2$

Fot. 2 — Oscylogram sygnału odczytanego z taśmy C-60 prod. Stilon Gorzów, dla średniego położenia regulatora siły głosu, Y — 1V/dz, X — 200 μs/dz: a — dla środkowego położenia regulatora barwy; b — dla maksymalnie stłumionych tonów wysokich; c — dla maks. stłumionych tonów niskich



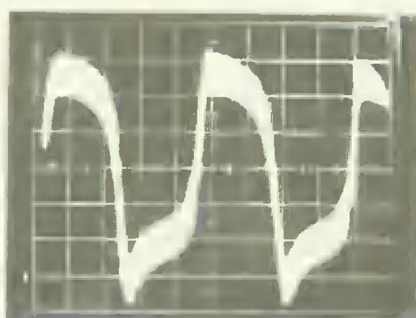
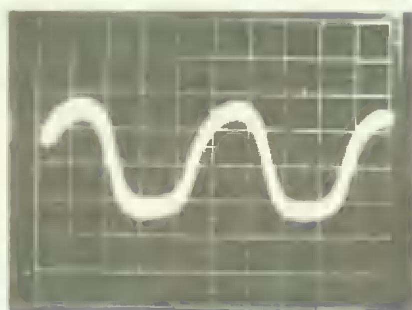
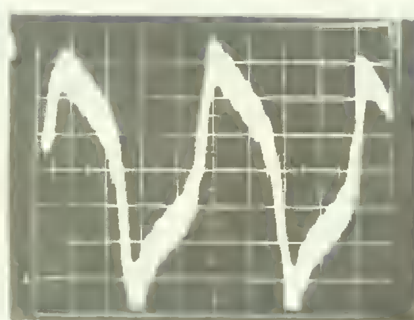
T1	O	Z
E	–	0,1
B	0,6	0,68
C	0,85	0,9

T2	O	Z
E	2,3	2,3
B	2,8	2,8
C	6,6	6,6

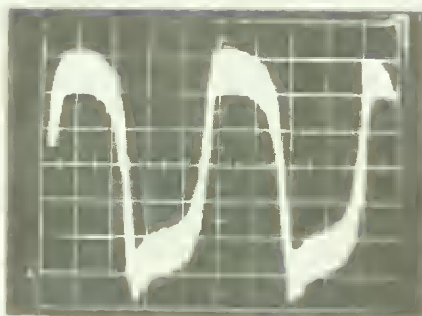
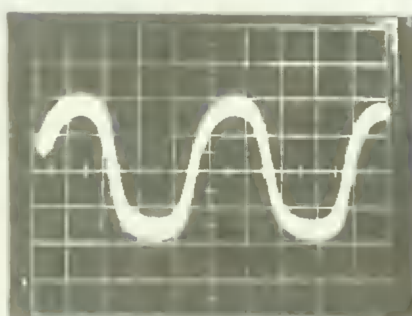
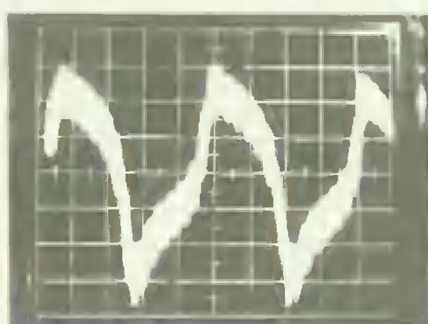
T3	O	Z
E	7,4	7,4
B	6,75	6,75
C	3,4	3,4



Rys. 2 — Schemat urządzenia kasetowego MK433 Data Recorder

Fot 3 — Jak fot. 2, lecz dla tasmy Stilon Gorzow C-90

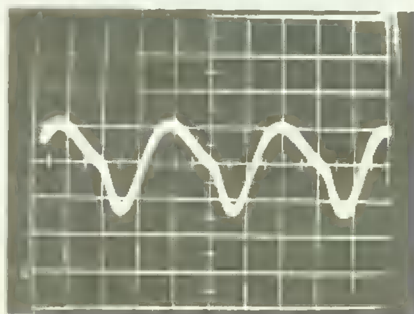




Fot. 4 — Jak fot. 2, lecz taśma C-90 żelazowa, produkcji zachodniej (średniej jakosci)

zane są ■ bardzo dobrą głowicą zastosowaną w MK433 — typu U31-34-2, specjalnie selekcjonowaną dla tego wyrobu. ■ także dokładną regulacją mechanizmu prowadzenia taśmy.

Niestety, urządzenie przekazane redakcji nie posiadało schematu ideowego, instrukcja eksploatacji również była wydrukowana w sposób raczej tymczasowy i mało komunikatywny. Praktycznie schemat MK433 jest niemal identyczny, jak MK432 — jednak ze względu na trudności w jego zdobyciu publikujemy go (za zgodą ZWM Lubartów) poniżej — może to być wskazówka dla tych wszystkich, którzy chcą przerobić posiadany MK432 na MK433, dodając przełącznik blokujący automatykę nagrywania. Należy przy tym pamiętać, że sygnał wyjściowy ■ komputera ZX Spectrum dołączamy do nóżki nr 3 gniazda wejściowego, połączonej z odpowiednim dzielnikiem dopasowującym amplitudę sygnału do czułości wzmacniacza. Przeróbka taka, co prawda, nie gwarantuje osiągnięcia tak dobrych parametrów, jak w oryginalnym MK433 (selekcjonowana głowica, dokładna regulacja mechanizmu), lecz za to za około dwukrotnie niższą cenę (podobno MK433 już nie jest urządzeniem powszechnego użytku i dlatego...).

Grzegorz Zaloi



Fot. 5. — Przydźwięk układu odchylania poziomego monitora ustawionego 30 cm od MK433, Y — 500 mV dz, X — 20 μs dz

c.d. z II str. okł

wanej wersji któregoś ■ programow, wprowadzając stopniowo rożne modyfikacje. Poszczególne sprawdzone rozwiązania cząstkowe nadają się do zastosowania we własnych programach. Niektore z zawartych ■ dyskietce programow mogą miec zresztą sporą wartość użytkową.

Książka zasługuje ■ duże uznanie zarówno za niekonwencjonalną formułę, jak i z powodu jej dobrego dopasowania do oczekiwań licznej grupy użytkowników komputerów, preferujących przeplatanie lektury ■ praktycznymi eksperymentami i ćwiczeniami przy klawiaturze. Czy doczekamy się kiedys takich książek w języku ojczystym? (rw)

CBASIC — DAN EINFUHRUNGS- UND NACHSCHLAGEBUCH FUR ANWENDER. W. Egerichs R. Weiss, Huthig Verlag, Heidelberg 1985, 172 strony.

Język programowania CBASIC jest dialektem BASIC-a, przeznaczonym przede wszystkim do zastosowań w administracji, księgowości i przetwarzaniu danych. W związku z tym CBASIC zawiera dość efektywne narzędzia do organizacji zbiorów dyskowych oraz do operacji ■ tekstach, jak np. cenna przy porównywaniu tekstów funkcja tekstowa UCASE, zamieniająca wszystkie małe litery występujące w jej argumencie na duże. Podstawowe elementy języka CBASIC przypominają jednak popularny MICROSOFT-BASIC.

CBASIC jest lansowany przez firmę Digital Research, twórcę systemu operacyjnego CP M. Nic dziwnego, że CBASIC jest dostępny we wszystkich jego mutacjach: CP M-80, CP M-86, MP M, Concurrend CP M itd.

Książka pt. „CBASIC" wprowadza Czytelnika w arkana programowania w tym języku. Kolejno omawiane są: typy danych, wyrażenia i funkcje, instrukcje wejścia wyjscia, przypisania, rozkazy sterujące przebiegiem programu, podprogramy, formatowane operacje wyjscia, organizacja i obsługa zbiorów w pamięci masowej. Jeden ■ ostatnich rozdziałów poświęcony jest w całości kompletnemu przykładowi zastosowań, prostej bazie danych.

Wykład jest przystępny, komunikatywny, tresciwy, metodyczny i bogato ilustrowany zarowno diagramami składniowymi instrukcji, przykładami gotowych programow, jak i ich struktogramami. Autorzy postanowili naiwyraźniej ukazać CBASIC jako narzędzie pozwalające na porządne, strukturalne programowanie i cel ten w dużej mierze osiągnęli. Wskazane i objaśnione zostały rożnice między interpreterem, ■ kompilatorem. Autorzy zasygnalizowali także rozbieżności między językiem CBASIC, a innymi dialektami BASIC-a. Dla użytkowników kompilatora zamieszczono też krotki opis zasad korzystania z typowego dla systemu CP/M edytora tekstów ED.

Reasumując: dobry podręcznik języka CBASIC, pozwalający na samodzielne opanowanie wszystkich jego możliwości, przeznaczony jednak dla osób mających pierwszy kontakt z BASIC-em już za sobą. Mankamentem książki jest brak skorowidza rozkazów, co utrudnia posługiwanie się nią w bieżącej pracy przy komputerze. (rw)

ASEMBLER — GENS 3

Część [illegible]

1. Porównanie asemblera z innymi językami programowania.

Wyjaśnijmy na wstępie kilka pojęć, których interpretacja może byc niejasna przy pierwszym czytaniu tekstu.

Słowo „asembler" jest często używane do określania dwu różnych pojęć. W pierwszym znaczeniu będzie oznaczało pewien język programowania, posiadający właściwą sobie składnię i w szczególności ściśle określony zbiór instrukcji zdeterminowany rodzajem komputera. W drugim znaczeniu będzie oznaczało pewien konkretny program (np. „Gens3" dla mikrokomputera Spectrum) umożliwiający tworzenie programów zgodnie z regułami języka programowania.

Powszechnie stosowane języki programowania jak Algol, BASIC, Cobol, Fortran, Pascal, posiadają tzw. raporty (definicje pewnego wzorca języka). Algorytmy w nich napisane są w dużym stopniu niezależne od konkretnego komputera. Różnice konkretnych wersji i realizacji dotyczą głównie repertuaru instrukcji wejścia i wyjścia. Można sobie wyobrazić sytuację, że po zapoznaniu się z nimi uda nam się zrozumieć program napisany na inny komputer i w innym języku od tego, którym się posługujemy i jest zrozumiały dla naszego komputera. W końcu uruchomimy poprawnie działającą wersję interesującego nas programu. Oczywiście w wielu przypadkach taka translacja może nie mieć żadnego praktycznego zastosowania. Przykładowo mija się z celem pisanie obliczeń numerycznych w Cobolu lub przetwarzania danych w Algolu, ponieważ języki te stworzono z myślą o dokładnie przeciwnych zastosowaniach. Oczywiście można rozbudować dany język do tego stopnia, że będzie się nadawał prawie do wszystkich zastosowań. Niestety jest to okupione m. in. czasem jego poprawnego opanowania, a często i przejrzystości, wszystkich jego struktur.

Zupełnie odmienna jest idea programowania w asemblerze. O ile tworzenie programu w językach wyższego rzędu przypomina wznoszenie konstrukcji budowlanej z wielu uniwersalnych elementów wielkopłytowych, to w asemblerze dysponujemy w chwili rozpoczęcia budowy wyłącznie surowcem do produkcji elementarnych cegiełek. Ponadto w wymyslaniu ewentualnych elementow przyspieszających tempo ukonczenia budowli i jej estetyki mamy prawie całkowitą swobodę.

Niezależnie od użytego języka programowania każdy program jest ostatecznie realizowany przez procesor. Niestety, lista zleceń, które jest on w stanie zrozumieć i wykonać jest na ogół dość odległa od zadania, które zamierzamy rozwiązać. Problem tłumaczenia danego języka programowania na język rozkazów zrozumiałych dla procesora jest realizowany różnymi sposobami.

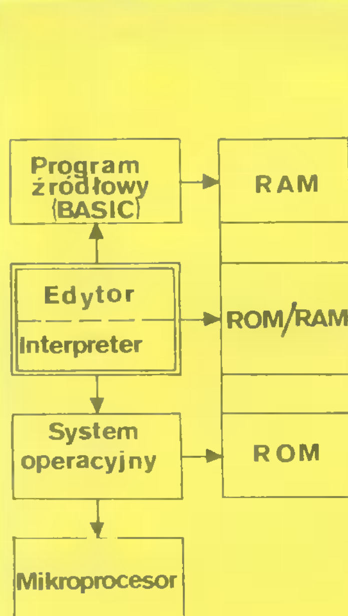
W przypadku mikrokomputerów najczęściej stosowanym rozwiązaniem jest istnienie wbudowanego na stałe programu interpretera. Wykonanie aktualnie zapisanego w pamięci programu polega na jego ciągłym tłumaczeniu przez interpreter. Każda przetłumaczona instrukcja jest realizowana przez wykonanie sekwencji rozkazów elementarnych przez mikroprocesor.

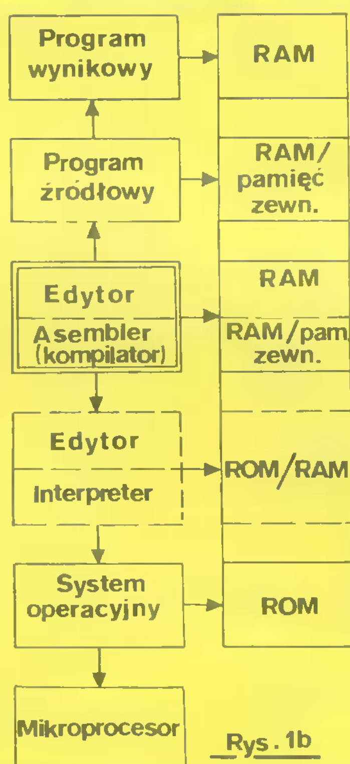
Drugim rozwiązaniem jest jednorazowe przetłumaczenie programu przy pomocy dodatkowego programu tzw. kompilatora na ciąg rozkazów, które może bezpośrednio realizować mikroprocesor. Tak otrzymaną postać programu będziemy nazywać programem wynikowym, a jego pierwotną postać programem źródłowym.

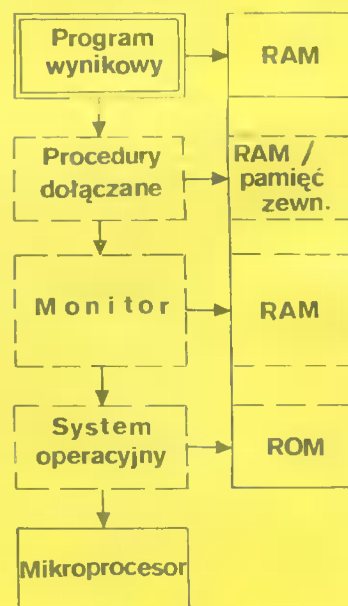
Należy tu podkreślić, że realizacja programu wynikowego nie wymaga obecności w pamięci programu źródłowego, ani (przeważnie) kompilatora. Różnice obu rozwiązań ilustruje rys. 1.

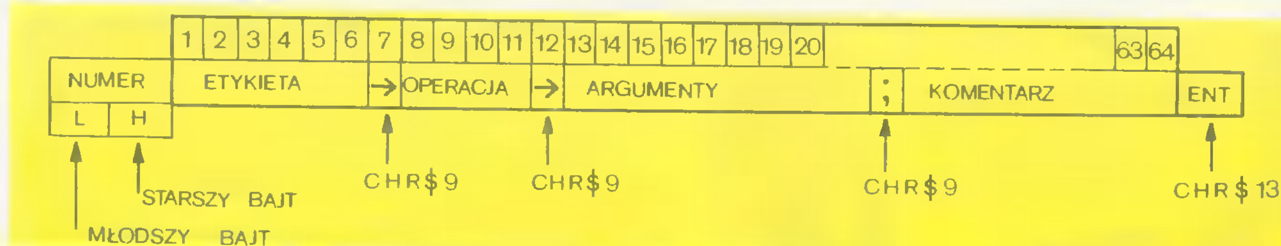
Zaletą pierwszego rozwiązania jest możliwość natychmiastowego wykonania

Rys. 1. a — typowa konfiguracja pamięci dla programów interpretowanych, b — konfiguracja pamięci dla programów kompilowanych w fazie, c — konfiguracja pamięci dla programów kompilowanych w fazie testowania i realizacji programu









Rys. 2. Format linii programowej w języku asemblera

wpisanego programu, a wadą wolny czas realizacji i reguły większa objętość programu. Ponadto dla programowania w asemblerze charakterystyczne jest wyłącznie drugie rozwiązanie. Wynika to z faktu, że pojedyńcze instrukcje tego języka odpowiadają w istocie rozkazom mikroprocesora, a zadaniem programu tłumaczącego jest głównie dostarczenie środków ułatwiających efektywne tworzenie programu. Opisany wcześniej edytor będący integralną częścią asemblera „Gens3" umożliwi nam redagowanie wersji źródłowej programu. W następnym etapie asemblacji najważniejszymi środkami pomocniczymi są nazwy mnemoniczne kodów instrukcji i nazwy adresów symbolicznych (etykiet).

Wybór konkretnego asemblera dla ilustracji wykładu, podyktowany względami dydaktycznymi, zmniejsza w nieznacznym stopniu ogólność wykładu. Większość omawianych poniżej problemów jest bowiem typowa dla innych asemblerów procesora Z80.

II. Składnia języka asemblera.

Program źródłowy w asemblerze składa się z ponumerowanych narastająco linii programowych. Numery linii mają znaczenie wyłącznie porządkowe. Linia programu jest podzielona na cztery pola: 1) Pole etykiety, 2) Pole operacji, 3) Pole argumentow, 4) Pole komentarza.

W standardowej postaci, w jakiej jest zapamiętywany wprowadzany program każde wypełnione pole jest poprzedzane znakiem tabulacji, a cała linia zakończona znakiem CHR$(13) ENTER. W jednej linii mogą wystąpić maksymalnie 64 znaki.

Wypełnienie wszystkich pól nie jest konieczne. W szczególności dopuszcza się istnienie następujących linii:

Linii Pustej — (zawierającej numer i jeden znak tabulacji), dla większej czytelności programu,

Linii Komentarza — (zawierającej numerem znak średnika, a następnie dowolny tekst),

Linii bez etykiety — (użycie etykiet wynika ze struktury programu).

Pojedynczą linię programową będziemy dalej nazywali zdaniem asemblera. Wyróżnimy kilka rodzajów zdań asemblera:

1) Generujące Kod. 1.1. Instrukcje Programowe; 1.2. Pseudoinstrukcje rezerwacji pamięci.

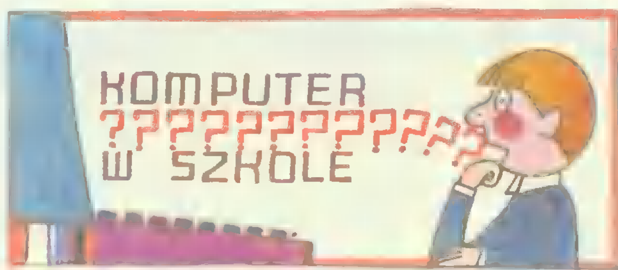
2) Nie generujące kodu, 2.1. Pseudoinstrukcje asemblacji, 2.2. Dyrektywy asemblacji.

1.1. Instrukcje programowe

Instrukcje programowe stanowią zasadniczy element asemblera. Mnemonik odpowiadający danemu rozkazowi mikroprocesora umieszczony jest w polu operacji, a wymagane argumenty w polu argumentów. Wszystkie mnemoniki są dwu-, trzy- lub czteroliterowe. Jeśli w danym rozkazie występują dwa argumenty, to oddzielamy je znakiem przecinka. Mnemoniki rozkazów oraz nazwy warunków i rejestrów procesora są pamiętane w wewnętrznym słowniku nazw asemblera i jako słowa kluczowe nie powinny być stosowane do innych celów (np. tworzenie nazw etykiet). Asembler dopuszcza jednak użycie nazw rozkazów i pseudoinstrukcji do tych celów, natomiast użycie nazwy rejestrów, warunków oraz znaku $ zostanie potraktowane jako błąd składni i uniemożliwi asemblację. Poniżej podajemy kompletną listę słów zastrzeżonych oraz nazw rozkazów:

1. Słowa zastrzeżone:
   Nazwy rejestrów: A B C D E H L I R AF AF' BC DE HL IX IY SP
   Nazwy warunków: M NC NZ PE PO Z
   Nazwa licznika lokacji: $ (znak dolara)

```
10 ;PROGRAM "SZACHOWNICA"
20 ;------------------------------
30 ;ETYKIETY
40 ;------------------------------
50 ATRYBU EQU   22528     ;de
60 KANAL  EQU   #1601     ;he
70 CLS    EQU   #0D6B
80 PRINT  EQU   #203C
90 AT     EQU   #0009
100 ;------------------------------
110 ;STAŁE
120 ;------------------------------
130 LINII  EQU   24
140 KOLUMN EQU   32
150
160 ;------------------------------
170 ;LOKACJA
180 ;------------------------------
190        ORG   60000
200        JR    START
210
220 ;------------------------------
230 ;DEKLARACJE ZMIENNYCH
240 ;------------------------------
250 JASNE  DEFB %01111000
260 ;bright 1, paper 7, ink 0
270
280 CIEMNE DEFB %00000111
290 ;bright 0, paper 0, ink 7
300
310 TEKST  DEFM "SZACHOWNICA"
320 DLUGOS DEFW 11
330 POZYCJ DEFB 32,22
340 ;at 1,2
350
360 ;------------------------------
370 ;PROGRAM GŁOWNY
380 ;------------------------------
390 START  CALL CLS
400        LD   A,2
410        CALL KANAL
420
430        LD   BC,(POZYCJ)
440        CALL AT
450        LD   DE,TEKST
460        LD   BC,(DLUGOS)
470        CALL PRINT
480
490        LD   HL,ATRYBU
500        LD   B,LINII/2
510
520 NASTEP PUSH BC
530        LD   B,KOLUMN
540        CALL PARZYS
550        LD   B,KOLUMN/2
560        CALL NIEPAR
570        POP  BC
580        DJNZ NASTEP
590        RET
600 ;------------------------------
610 ;PODPROGRAMY
620 ;------------------------------
630 PARZYS LD   A,(JASNE)
640        LD   (HL),A
650        INC  HL
660        LD   A,(CIEMNE)
670        LD   (HL),A
680        INC  HL
690        DJNZ PARZYS
700        RET
710
720 NIEPAR LD   A,(CIEMNE)
730        LD   (HL),A
740        INC  HL
750        LD   A,(JASNE)
760        LD   (HL),A
770        INC  HL
780        DJNZ NIEPAR
790        RET
800 ;------------------------------
```



POCHODNE FUNKCJI — INACZEJ

Przedstawiony tu program liczy już sobie parę lat, lecz nie stracił do dziś nic ze swojej atrakcyjności, udowadnia on bowiem, że dokonywanie symbolicznych obliczeń automatycznych nie jest tylko domeną drogich komputerów personalnych, ale może być także dostępne małym, domowym mikrokomputerom. Napisany w języku BASIC program umożliwia uzyskanie analitycznej postaci pochodnej dla dowolnej, nawet bardzo złożonej funkcji.

Program dokonuje w zasadzie wyłącznie operacji na tekstach. Rekurencyjna procedura (linia 170) przeprowadza rozbiór podanej funkcji w kolejności rosnących priorytetow operacji, aż do uzyskania pojedynczego wystąpienia zmiennej „x" i stałych. Następnie, na podstawie wartości zgromadzonych w tablicach, budowana jest pochodna funkcji. Po otrzymaniu wyniku można prześledzić kolejne etapy analizy i syntezy wyświetlając zawartości tablic f$ i g$.

Obsługa programu jest prosta. Po uruchomieniu przez RUN, po pytaniu „f(x) " należy podać funkcję, dla której chcemy uzyskać pochodną (przykłady na wydruku). Nazwy funkcji elementarnych należy pisać małymi literami, zaś ich argumenty umieszczać w nawiasach. Pochodna wyznaczana jest dla zmiennej x — pozostałe symbole traktowane są jak stałe. Program sygnalizuje błędy wyświetlając zadaną funkcję do znaku, który błąd spowodował. Zwykle jest nim spacja lub nadmiarowy nawias. Należy wowczas jeszcze raz napisac całe wyrażenie.

Podanie po pytaniu „f(x) =" znaku „?" powoduje wyswietlenie nazw funkcji, dla których program wyznacza pochodne. Naciśnięcie samego ENTER zatrzymuje program. Lista funkcji i ich pochodnych może być łatwo modyfikowana. Pojedyncza liczba w linii 4010 oznacza liczbę definiowanych funkcji.

Przydatność programu ocenią ewentualni użytkownicy, teraz wskażemy pewne jego wady.

W przypadku złożonych funkcji czas oczekiwania na wynik wynosi parę minut. Program nie jest więc rewelacyjnie szybki. Nie rozpoznaje on także jednoargumentowego operatora zmiany znaku. Np. dla funkcji sin(- -x) należy użyć mało eleganckiego zapisu (-1) * sin(0- -x). Trzecia niedogodność związana jest ze specyfiką BASICa ZX Spectrum. Program operuje na tekstach przechowywanych w tablicach. Taka struktura danych wymaga podania ograniczenia długości pamiętanego tekstu. Napis-pochodna jest zwykle znacznie dłuższy od napisu-funkcji i może się zdarzyć, że przekroczy on owo ograniczenie. Nadmiarowe znaki zostaną wówczas zignorowane. Wyjątkowo złośliwy to przypadek, gdy wynik z pozoru jest poprawny, faktycznie zaś brak jednego ze składników.

Rozmiary tablic tekstowych można dopasować do własnych wymagań. W linii 40 zmienna m określa dopuszczalną liczbę funkcji wewnętrznych, przyjętą raczej na wyrost. Zmniejszając tę liczbę np. do 10 można znacznie zwiększyć długość tekstów w tablicach.

Większość funkcji rozpoznawanych przez program jest dostępna bezpośrednio na klawiaturze ZX Spectrum i jego użytkownicy mogą użyć ich w programie. Zmniejsza to ryzyko błędu w danych, skraca długość pamiętanych tekstów i przyspiesza działanie programu. Brakujące funkcje hiperboliczne można zapisać np. SIN H(x).

Przeniesienie programu na inny system nie powinno nastręczać trudności. Jest to jednak zajęcie pracochłonne. Dla tych, którzy mają na to ochotę, kilka uwag:

— Sinclair BASIC, w przeciwieństwie do innych dialektów, nie dopuszcza indeksu tablicy równego 0. Najmniejsza wartość indeksu wynosi tu 1. Tablicami są także napisy.

— Tablice znakowe zadeklarowane są jako dwuwymiarowe (linie 50 i 51). Drugi wymiar określa maksymalną, stałą długość napisu. Jeśli dany dialekt BASICa tego nie wymaga, drugi wymiar można pominąć. Zbędna jest wówczas procedura obcinająca spacje linia 5000 i używana przez nią zmienna t$.

— Program wykorzystuje wyniki operacji logicznych (np. w linii 450). Wyrażenie t 0 ma wartość 1 (prawda) lub 0 (fałsz). Microsoft BASIC przyjmuje —1 jako wartość logicznej prawdy. Konieczna może więc być zmiana znaku.

— Sinclair BASIC ma własny zapis operacji tekstowych, którym najczęściej odpowiadają funkcje:

a$(5)	MID$(a$,5,1)
a$(6 TO 9)	MID$(a$,6,3)
a$(2 TO)	MID$(a$,2)
a$(TO 4)	LEFT$(a$,4)
a$(LEN a$—2 TO)	RIGHT$($,2).

Poczciwy BASIC nie ma ostatnio najlepszej prasy. Programistyczni puryści mają więc okazję napisania podobnego programu w którymś z modnych obecnie języków, jak Pascal, C czy Logo. Rekurencyjny algorytm powinien być dla nich dodatkowym dopingiem.

(M.G.)

```
  10   REM SYMBOLICZNE OKRESLANIE
             POCHODNYCH FUNKCJI
  40 LET m=16
  50 DIM f$(m,100) DIM g$(m,100): DIM u$(m,100)
DIM v$(m,100): DIM w$(m,100)  DIM q(m)· LET z$="+
*/^"
  51 RESTORE   READ fz. DIM x$(fz,8). DIM y$(fz,15
)
  52 FOR i=1 TO fz
  53 READ x$(i),y$(i)
  55 NEXT i
  60 LET z=1
  70 INPUT "f(x)=": LINE t$: IF t$="" THEN  GO TO
9999
  71 IF t$="?" THEN  GO TO 8000'
  72 LET f$(1)=t$
  75 PRINT #0;AT 0,9; FLASH 1;"PROSZE CZEKAC"
  80 GO SUB 170
  81 INPUT ""
  85 LET t$=f$(1)  GO SUB 5000
  90 PRINT "f(x) = ";t$
  92 LET t$=g$(1): GO SUB 5000
  95 PRINT "f'(x)= ";t$
 100 PRINT   GO TO 60
 110   REM
 120   REM POCHODNA FUNKCJI f(x)
 130   REM
 170 LET t$=f$(z). GO SUB 5000: LET e$=t$
```

```
 180 LET s$=e$: GO SUB 2010
 200 IF p=1 OR p=l THEN  GO TO 1000
 260 IF p=0 THEN  GO TO 700
 275   REM
 280   REM  f=u#v
 285   REM
 290 LET q(z)=o: LET u$(z)=e$( TO p-1): LET v$(z)=
e$(p+1 TO )
 300 LET z=z+1: LET f$(z)=u$(z-1): GO SUB 170: LET
w$(z-1)=g$(z)
 310 LET f$(z)=v$(z-1): GO SUB 170: LET z=z-1
 320 LET t$=u$(z): GO SUB 5000: LET c$=t$: LET t$=
v$(z): GO SUB 5000: LET d$=t$: LET t$=w$(z): GO SU
B 5000: LET a$=t$: LET t$=g$(z+1): GO SUB 5000: LE
T b$=t$: LET o=q(z)
 330 IF o=5 THEN  GO TO 540
 331 IF o=3 OR o=4 THEN  GO TO 400
 332 IF o=1 OR o=2 THEN  GO TO 350
 335   REM
 340   REM  f=u+v  lub  f=u-v
 345   REM
 350 IF a$<>"0" THEN  IF b$<>"0" THEN  LET g$(z)=a
$+z$(o)+b$: RETURN
 360 IF a$<>"0" THEN  LET g$(z)=a$: RETURN
 370 IF b$<>"0" THEN  LET g$(z)=z$(2 TO o-1)+b$: R
ETURN
 380 LET g$(z)="0": RETURN
 385   REM
 390   REM  f=u*v  lub  f=u/v
 395   REM
 400 IF a$="0" THEN  LET t=0: LET r$=a$: GO TO 440
 410 LET t=1
 420 IF a$="1" THEN  LET r$=d$: GO TO 440
 425 LET s$=a$: GO SUB 2000: IF o<3 THEN  LET a$="
("+a$+")"
 428 LET p$=d$: LET s$=p$: GO SUB 2000: IF o<3 THE
N  LET p$="("+p$+")"
 430 LET r$=a$+"*"+p$: GO TO 440
 440 IF b$="0" THEN  GO TO 480
 445 LET p$=a$: LET s$=a$: GO SUB 2000: IF o<3 THE
N  LET p$="("+p$+")"
 450 LET t=t+1: LET r$=r$(1+(t=0) TO ): LET a=q(z)
-2-((t=1) AND (q(z)=3)): LET r$=r$+z$(q(z)-2 TO a)
 451 IF r$="0" THEN  LET r$=""
 460 IF b$="1" THEN  LET r$=r$+c$: GO TO 480
 465 LET s$=c$: GO SUB 2000: IF o<3 THEN  LET c$="
("+c$+")"
 470 LET s$=b$: GO SUB 2000: IF o<3 THEN  LET b$="
("+b$+")"
 475 LET r$=r$+c$+"*"+b$
 480 IF r$="0" OR q(z)=3 THEN  GO TO 510
 490 IF t=2 THEN  LET r$="("+r$+")"
 495 LET s$=d$: GO SUB 2000: IF o<3 THEN  LET d$="
("+d$+")"
 500 LET r$=r$+"/"+d$+"^2"
 510 LET g$(z)=r$
 520 RETURN
 525   REM
 530   REM  f=u^v
 535   REM
 540 IF b$<>"0" THEN  GO TO 610
 550 IF a$="0" THEN  LET g$(z)=a$: RETURN
 560 IF a$="1" THEN  LET r$="": GO TO 590
 570 LET s$=a$: GO SUB 2000: IF o<3 THEN  LET a$="
("+a$+")"
 580 LET r$=a$+"*"
 590 IF d$<>"1" THEN  LET r$=d$+"*"+r$
 595 IF d$(1)="(" THEN  LET d$=d$(2 TO )
 596 GO SUB 6000
 597 IF b$(1)=" " THEN  LET b$=b$(2 TO ): GO TO 59
9
 598 LET b$="("+b$+")"
 599 IF b$="1" THEN  LET g$(z)=r$+c$: RETURN
 600 IF b$="0" THEN  LET g$(z)=r$+"1": RETURN
 605 LET g$(z)=r$+c$+"^"+b$: RETURN
 610 LET p$=c$
 620 LET s$=p$: GO SUB 2000: IF o<3 THEN  LET p$="
("+p$+")"
 630 LET s$=a$: GO SUB 2000: IF o<3 THEN  LET a$="
("+a$+")"
 640 LET s$=d$: GO SUB 2000: IF o<3 THEN  LET d$="
("+d$+")"
 650 LET s$=b$: GO SUB 2000: IF o<3 THEN  LET b$="
("+b$+")"
 660 LET t$=f$(z): GO SUB 5000: LET r$=t$+"*(": IF
 a$<>"0" THEN  LET r$=r$+a$+"/"+p$+"*"+d$+"+"
 665 LET r$=r$+"ln"
 666 IF c$(1)<>"(" THEN  LET c$="("+c$+")"
 667 LET r$=r$+c$
 670 IF b$="1" THEN  LET g$(z)=r$+")": RETURN
 680 LET g$(z)=r$+"*"+b$+")": RETURN
 685   REM
 690   REM  f=f(u)
 695   REM
 700 FOR i=1 TO fz
 705 LET t$=x$(i): GO SUB 5000
 710 IF LEN e$>=LEN t$ THEN  IF e$( TO LEN t$)=t$
THEN  GO TO 720
 715 NEXT i: GO TO 910
 720 LET ii=i: IF e$(LEN e$)<>")" THEN  GO TO 1000
 725   REM
 730   REM  u'
 732   REM
 735 LET q(z)=ii
 740 LET z=z+1: LET f$(z)=e$(LEN t$+1 TO l-1): GO
SUB 170: LET z=z-1
 750 LET t$=f$(z): GO SUB 5000: LET e$=t$: LET t$=
f$(z+1): GO SUB 5000: LET c$=t$: LET t$=g$(z+1): G
O SUB 5000: LET a$=t$
 760 IF a$="0" THEN  LET g$(z)="0": RETURN
 770 LET s$=a$: GO SUB 2000: IF o<3 THEN  LET a$="
("+a$+")"
 790 LET o=q(z): IF o>10 THEN  GO TO 850
 795   REM
 800   REM  f'=u'*g(u)
 805   REM
 810 LET t$=y$(o): GO SUB 5000: LET r$=t$+c$+")":
IF a$<>"1" THEN  LET r$=a$+"*"+r$
 820 IF o>6 THEN  LET r$=r$+"^2)"
 830 LET g$(z)=r$: RETURN
 835   REM
 840   REM  f'=u'/g(u)
 845   REM
 850 LET t$=y$(o): GO SUB 5000: LET r$=t$+c$+")":
LET r$=a$+"/"+r$
 860 IF o<>19 THEN  LET r$=r$+"^2)"
 870 LET g$(z)=r$: RETURN
 895   REM
 900   REM  x, stale i nawiasy
 905   REM
 910 IF e$(1)<>"(" OR e$(LEN e$)<>")" THEN  GO TO
930
 915 LET s$=e$(2 TO LEN e$-1): GO SUB 3000: IF NOT
 n THEN  LET g$(z)="0": RETURN
 920 LET f$(z)=e$(2 TO LEN e$-1): GO TO 170
 930 IF e$="x" THEN  LET g$(z)="1": RETURN
 940 LET g$(z)="0": RETURN
1000 PRINT "? niepoprawne wyrazenie:"' INK 2;e$: I
NK 0: INPUT "": PRINT #1;"  Nacisnij dowolny klawi
sz !": PAUSE 0: CLEAR : RUN
2000   REM
2002   REM  ANALIZA FUNKCJI f(x)
2005   REM
2010 LET l=LEN (s$): LET p=0: LET o=6: LET kl=0
2030 FOR i=1 TO l
2040 LET k$=s$(i)
2050 LET kl=kl-(k$=")"): LET kl=kl+(k$="(")
2060 IF kl<>0 THEN  GO TO 2100
2070 IF k$="+" OR k$="-" THEN  LET p=i: LET o=1+(k
$="-")
2080 IF k$="*" OR k$="/" THEN  IF o>2 THEN  LET p=
i: LET o=3+(k$="/")
2090 IF k$="^" THEN  IF o>4 THEN  LET p=i: LET o=5
2100 NEXT i
2110 IF kl THEN  GO TO 1000
2130 RETURN
3000   REM
3005   REM  TEST LICZBA/WYRAZENIE
3010   REM
3030 FOR i=1 TO LEN s$
3040 LET k$=s$(i)
3050 IF k$>"/" THEN  IF k$<":" THEN  NEXT i: LET n
=0: RETURN
3060 IF k$="-" THEN  NEXT i: LET n=0: RETURN
3070 LET n=1: RETURN
3995   REM
4000   REM  FUNKCJE I POCHODNE
4005   REM
```

```
4010 DATA 19
4020 DATA "sin(","cos("
4030 DATA "cos(","(-1)*sin("
4040 DATA "sinh(","cosh("
4050 DATA "cosh(","(-1)*sinh("
4060 DATA "exp(","exp("
4070 DATA "sqr("," 5/sqr("
4090 DATA "tan(","(1+tan("
4100 DATA "cot(","(-1)*(1+cot("
4110 DATA "tanh(","(1-tanh("
4120 DATA "coth(","(-1)*(1-coth("
4130 DATA "arcsin(","sqr(1-("
4140 DATA "arccos(","(-1)/sqr(1-("
4150 DATA "arctan(","(1+("
4160 DATA "arccot(","(-1)/(1+("
4170 DATA "arsinh(","sqr(1+("
4180 DATA "arcosh(","(-1)/(sqr(-1+("
4190 DATA "artanh(","(1-("
4200 DATA "arcoth(","(1-("
4210 DATA "ln("."("
4995   REM
5000   REM  OBCIECIE SPACJI
5005   REM
5020 FOR h=1 TO LEN t$: IF t$(h)<>" " THEN  NEXT h
: RETURN
5030 LET t$=t$( TO h-1): RETURN
5995   REM
6000   REM  x^(liczba)
6005   REM
6010 LET d=LEN d$: IF d$(d)=")" THEN  LET d$=d$( T
O d-1)
6020 FOR h=1 TO LEN d$: IF d$(h)>="A" AND d$(h)<="
Z" OR d$(h)>="a" AND d$(h)<="z" THEN  LET b$=d$+"-
1": RETURN
6030 NEXT h: LET b$=STR$ (VAL d$-1): RETURN
7995   REM
8000   REM   ? - LISTA FUNKCJI
8002   REM
8005 CLS : PRINT TAB 9; BRIGHT 1;"LISTA FUNKCJI":
PRINT : RESTORE 4020: FOR i=1 TO fz: READ a$,b$: P
RINT TAB 12;a$+"x)": NEXT i
8100 PRINT #0;"   Nacisnij dowolny klawisz": PAUSE
 0: RUN
9000   REM
9005   REM (C) T.Gawlick,  "mc" 2/84
9010   REM
9999 STOP
```

```
f(x) = T*sin(x)/x
f'(x)= (T*cos(x)*x-T*sin(x))/x^2

f(x) = a*sin(b*x+c)
f'(x)= a*b*cos(b*x+c)

f(x) = ln(x^3)+tanh(sqr(x-12.6))
f'(x)= 3*x^(2)/(x^3)+.5/sqr(x-12.6)*(1-tanh(sqr(x-
12.6))^2)

f(x) = exp((x^2+x)/cos(x))
f'(x)= ((2*x^(1)+1)*cos(x)-(x^2+x)*(-1)*sin(x))/co
s(x)^2*exp((x^2+x)/cos(x))
```

c.d. ze str. 29

2. Nazwy rozkazów:
ADC ADD AND BIT CALL CCF CP CPD CPDR CPI CPIR CPL DAA DEC DI DJNZ EI EX EXX HALT IM IN INC IND INDR INI INIR JP JR LD LDD LDDR LDI LDIR NEG NOP OR OTDR OTIR OUT OUTD OUTI POP PUSH RES RET RETI RETN RL RLA RLC RLCA RLD RR RRA RRC RRCA RRD RST SBC SCF SET SLA SRA SRL SUB XOR

Podstawową zmienną, której używa asembler w trakcie tłumaczenia programu jest licznik lokacji. Zmienna ta jest liczbą całkowitą z przedziału [0,65535]. Wyznacza ona adresy, pod którymi będą umieszczane kody rozkazów procesora odpowiadające kolejno tłumaczonym liniom programowym. Problem ten jest istotny z następujących powodów:

Rozkazy procesora mają różną długość (od jednego do czterech bajtów). Ponadto asembler musi nam umożliwić rezerwację dowolnej (ale sensownej) długości zmiennych oraz umożliwić mnemoniczne określenie adresów instrukcji, bowiem numery linii programowych mają jedynie znaczenie porządkowe w trakcie redagowania programu. Należy tu zaznaczyć, że oczywiście możliwe są programy, w których jedynymi zmiennymi będą rejestry procesora, ale na ogół będzie to niewystarczające.

Problem ten w asemblerze rozwiązują:

1.2. Pseudoinstrukcje rezerwacji pamięci

Nazwy pseudoinstrukcji i ich argumenty umieszczamy identycznie, jak w przypadku instrukcji programowych w polu operacji i argumentów. Nazwy i działanie pseudoinstrukcji są następujące:

DEFB — Umożliwia wypełnienie kolejnych komórek pamięci wyrażeniami jednobajtowymi. Kolejne wyrażenia oddzielamy znakiem przecinka.

DEFW — Umożliwia wypełnienie dwóch kolejnych komórek pamięci wyrażeniem dwubajtowym. Mniej znaczący bajt wyrażenia zostanie wpisany jako pierwszy.

DEFM — Umożliwia wypełnienie kolejnych komórek pamięci kodami ASCII odpowiadającymi znakom wpisanego w polu argumentów łańcucha. Łańcuch znaków powinien rozpoczynać się i kończyć znakiem apostrofu.

DEFS — Umożliwia rezerwację obszaru pamięci o długości do maksymalnie 65535 bajtów. Wartości zarezerwowanych bajtów nie są określone (obszar ten nie jest zerowany).

Liczba argumentów DEFB oraz długość łańcucha w DEFM jest ograniczona długością linii programowej. Po przetłumaczeniu tych pseudoinstrukcji zawartość licznika lokacji jest przez asembler zwiększana o odpowiednią wartość. Nazwy pseudoinstrukcji są pamiętane w wewnętrznym słowniku nazw, lecz nie są, podobnie jak nazwy instrukcji, zastrzeżone.

Nazwy instrukcji i pseudoinstrukcji nie są jedynymi nazwami dostępnymi w asemblerze. Jak już wspomniano w opisie edytora, asembler rezerwuje pewien obszar (jego wielkość określamy przed rozpoczęciem asemblacji) na nazwy używane przez programistę. Obszar ten nazywamy Tablicą Symboli. Jest on głównie wykorzystywany do przechowywania nazw etykiet. W trakcie asemblacji zapamiętanym tam etykietom są przyporządkowywane wyliczone wartości. Wartości te odpowiadają zawartości licznika lokacji w chwili analizowania linii programowej zawierającej etykietę. Programista może również nadawać konkretne wartości nazwom zdefiniowanym przez siebie, jak również wpływać bezpośrednio na wartość licznika lokacji. Służą do tego:

2.1. Pseudoinstrukcje asemblacji

ORG — umożliwia ustalenie początkowej wartości licznika lokacji (lub jego ewentualną zmianę). Wymaganą wartość umieszczamy w polu argumentów.

EQU — umożliwia nadanie wybranej przez nas nazwie określonej wartości. Nazwę umieszczamy w polu etykiety, mnemonik w polu operacji, a żądaną wartość liczbową w polu argumentów.

Nazwa użyta przez programistę może składać się z następujących znaków: małe i duże litery, cyfry oraz znaki specjalne $, ., [,], ↑, __, £.

Pierwszym znakiem musi być litera, ponadto nazwa nie może zawierać wewnątrz znaku spacji.

Tadeusz Basista

DYSKIETKA

W poprzednim numerze, opisana była stacja miękkich dysków 5 1/4 cala. Dyskietki tej średnicy są obecnie jednym z najpopularniejszych nośników informacji, stosowanych w minikomputerach. Konstrukcja dyskietki została uproszczona do niezbędnego minimum, udało się jednak, dzięki odpowiedniej technologii, zachować doskonałe parametry mechaniczne i magnetyczne. W dobrej klasy dyskietkach podłoże wykonane jest z folii mylarowej, produkowanej przez firmę Du Pont. Folia ta wykazuje bardzo małą wrażliwość na zmiany temperatury i wilgotności otoczenia.

Materiał magnetyczny nie różni się zasadniczo, od materiału w zwykłych taśmach magnetofonowych (używa się mieszaniny tlenków żelaza i chromu) jednak stosuje się specjalne techniki obróbki, mające na celu uzyskanie kryształków nośnika odpowiednio ukształtowanych i o odpowiedniej granulacji. Ważne są nie tylko wymiary poszczególnych kryształków związku magnetycznego, ale również ich zorientowanie w przestrzeni. Chodzi o to, aby pole magnetyczne głowicy zapisującej, działało względem osi najłatwiejszego magnesowania. Związki te są silnie anizotropowe i wykazują różne parametry magnetyczne, w zależności od kąta pod jakim są magnesowane. Złe zorientowanie kryształków może znacznie obniżyć pozostałość magnetyczną, uzyskaną w wyniku zapisu informacji na taśmie. Dzięki stosowaniu specjalnych technologii i bardzo dokładnej kontroli nośnika, udało się prawie całkowicie wyeliminować tzw. efekty drop-out, czyli dziury w nośniku magnetycznym, na których nie można dokonać zapisu ani odczytu informacji. Większość produkowanych obecnie dyskietek posiada w miejscu mocowania naklejone specjalne pierścienie wzmacniające, mające zapobiegać mechanicznym deformacjom podłoża i wyrównywać rozkład naprężeń, wynikający z nierównomiernego nacisku wrzeciona mocującego.

Ponieważ odsunięcie głowicy od nośnika na odległości rzędu zapisywanej fali obniża poziom sygnału kilkakrotnie, powierzchnia dyskietek powinna być całkowicie wolna od pyłu. W celu uniknięcia zapylenia powierzchni, dyskietka jest wyposażona w specjalną plastikową kopertę, pokrytą od wewnątrz miękkim materiałem o strukturze włóknistej, posiadającej bardzo duże możliwości wiązania na swej powierzchni drobin kurzu. Materiał ten ma właściwości antystatyczne, więc umożliwia usunięcie z powierzchni dyskietki ładunków elektrostatycznych, gromadzących się tam na skutek tarcia dyskietki o głowicę i kopertę. Największe gęstości zapisu uzyskuje się obecnie, stosując poprzeczne magnetyzowanie nośnika, uzyskane przy zastosowaniu głowic wykonanych technologią grubowarstwową. Jak dotąd jednak dominują standardowe głowice podobne do głowic stosowanych w magnetofonach, jednak o znacznie węższym śladzie: od setnych części milimetra do ok. 0,3 mm. Głowice te posiadają płaskie czoło, a szerokość szczeliny nie przekracza kilku μm.

Cały dysk podzielony jest na sektory, przy czym liczba sektorów może się zmieniać w zależności od odległości osi obrotu. Sektory mieszczą rekordy, zawierające 128, 256, 512 do 4 kBajty. Przed każdym sektorem znajduje się 4-bajtowy nagłówek informujący o czytanej stronie (w przypadku napędu dysku, wyposażonego w dwie głowice), numerze czytanej ścieżki, numerze czytanego rekordu oraz ilości bajtów danego rekordu. Za tą częścią informacyjną, znajduje się niewielka przerwa, po której są zapisane informacje wchodzące w skład właściwego rekordu. Na końcu sektora, zapisywana jest suma kontrolna, pozwalająca kontrolować na bieżąco jakość odczytu.

Do zapisu informacji na dyskietce stosuje się metodę, w której domeny magnetyczne nośnika, magnesowane są albo zgodnie z ruchem dyskietki, albo w stronę przeciwną bez powrotu do stanu pierwotnego nienamagnesowania. Nie ma potrzeby używania prądu podkładu w.cz., gdyż nie występuje konieczność linearyzacji charakterystyki magnesowania, z uwagi na występowanie tylko dwóch stanów namagnesowania nośnika. Do zapisu i odczytu, stosowana jest ta sama głowica, wyposażona najczęściej w odrębne uzwojenia. W czasie zapisu głowicę steruje się prądowo, aby osiągnąć jak najmniejsze stałe czasowe układu elektrycznego. Podczas odczytu, napięcie z głowicy podawane jest na szerokopasmowy wzmacniacz odczytu, po którym znajduje się komparator okienkowy.

Komparator okienkowy jest specjalnym układem elektronicznym, na którego wyjściu następuje zmiana stanu logicznego w momencie sygnał wejściowy „wchodzi" w określony przedział napięć. W dyskach, rejestruje się przechodzenie sygnałów w pobliżu zera. Na rysunku przedstawiono poglądowo, jak przebiega proces zapisu i odczytu informacji na nośniku informacji. Rysunek przedstawia zapis, w którym informacja kodowana jest w systemie FM. Poszczególne „komórki" na dyskietce, oddzielone są tutaj od siebie kolejnymi przemagnesowaniami nośnika magnetycznego w okresie T. Jeżeli wewnątrz „komórki" pojawi się przemagnesowanie, to kontroler dysku czyta je jako zapis logiczny „1". W czasie odczytywania logicznego „0", przemagnesowania pojawiają się tylko podczas przechodzenia z „komórki" do „komórki". Jest to konieczne w celu synchronizacji generatora znajdującego się w kontrolerze. Zapis FM jest metodą stosunkowo rozrzutną, gdyż ilość przemagnesowań nośnika jest znacznie większa od ilości bitów informacji zapisanych na dysku.

Znacznie oszczędniejsza pod tym względem jest metoda zapisu MFM. Metoda ta opiera się na modulacji położenia impulsu. Podczas odczytu ciągu jedynek, impulsy pojawiają się „na środku" każdej komórki, zawierającej jedynkę logiczną. Są to impulsy o dwukrotnie większej częstotliwości, co daje dwa razy większą gęstość zapisu niż w modulacji FM przy tej samej maksymalnej częstotliwości przemagnesowań podłoża. Jeżeli pojawia się pojedyncze zero logiczne, to wycinany jest jeden impuls z ciągu. W przypadku pojawienia się ciągu zer, pojawia się impuls po zakończeniu „komórki" zawierającej logiczne „0", jeżeli następuje po niej „komórka" zawierająca również logiczne „0".

Jeżeli następna „komórka" zawiera logiczną „1", to impuls pojawia się na środku tej „komórki". Taki rodzaj modulacji daje niezbędne przesunięcia fazowe impulsów, konieczne do odtworzenia zapisywanej informacji. Odtworzenie informacji następuje przez porównanie impulsów odczytywanych z dyskietki, z impulsami wytwarzanymi przez specjalny generator pracujący w pętli regulacji fazy, synchronizowany odczytywanymi impulsami. Ponieważ istnieją chwilowe wahania częstotliwości odczytywanych impulsów w zależności od aktualnie czytanej informacji, pętla regulacji fazy zawiera specjalny układ kluczujący, zapewniający stałą fazę impulsów generowanych przez generator, mimo zmian fazy impulsów synchronizujących.

W modulacji M²FM, w czasie odczytywania ciągu zer, impuls pojawia się tylko po odczytaniu pierwszego zera w ciągu. Sprawia to znaczne kłopoty z synchronizacją generatora w przypadku odczytywania rekordów, zawierających długie ciągi zer. Dlatego modulacja ta nie znalazła szerszego zastosowania w praktyce.

„Młody Technik — InforMik" wydaje Instytut Wydawniczy „Nasza Księgarnia"

Rada Redakcyjna: doc. dr Zygmunt Dąbrowski, inż. Jerzy Jasiuk, dr Zygmunt Kalisz, mgr Zbigniew Słowiński, mgr inż. Jerzy Siek, dr Zbigniew Płochocki, Piotr Postawka, mgr inż. Roland Wacławek, prof. dr hab. Andrzej K. Wróblewski (przewodniczący), mgr inż. Grzegorz Zalot.

Zespół redakcyjny: *„InforMik"* redaguje zespół *„Młodego Technika"*. Jerzy Klawiński (red. odpowiedzialny), Lidia Sadowska-Szlaga (red. techn.), Józef Trzionka (redaktor naczelny), Roland Wacławek (software), Grzegorz Zalot (hardware).

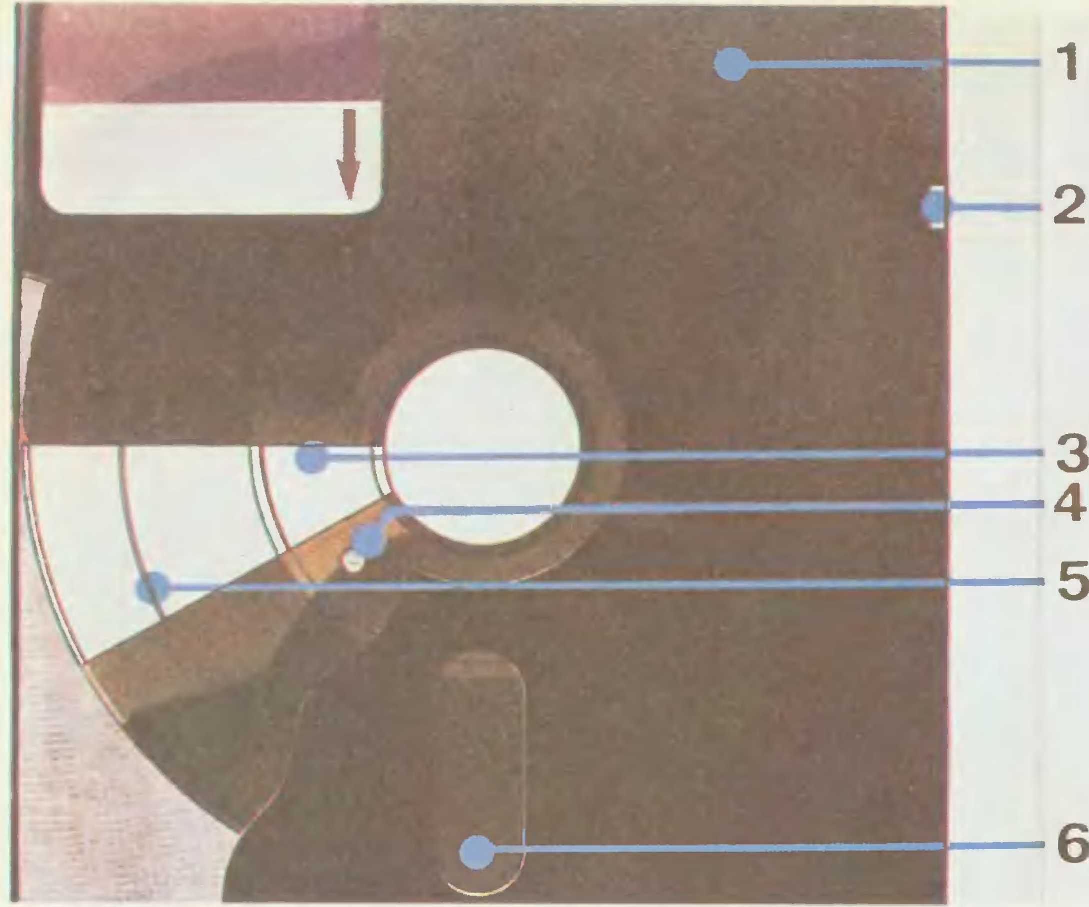
Stali współpracownicy: Wojciech Apel, Tadeusz Basista, Jacek Jędrzejowski, Piotr Postawka, Marek Szczepański, Krzysztof Wiśniewski.

Adres redakcji: ul. Spasowskiego 4, 00-389 Warszawa, lub skr. poczt. 380, 00-950 Warszawa. **Telefony:** centrala: 26-24-31 do 36. Dział Łączności z Czytelnikami — wewn. 60, pozostałe działy: wewn. 42 i 47. Redaktor naczelny: 26-26-27 lub wewn. 87.

Warunki prenumeraty: ogólnie obowiązujące w kraju. **W STAŁEJ SPRZEDAŻY INFORMIK JEST W SALONIE WYDAWNICZYM „NASZEJ KSIĘGARNI"** ul. Spasowskiego 4A.

Redakcja zastrzega sobie prawo adiustacji i skracania nadesłanych materiałów. Artykułów nie zamówionych redakcja nie zwraca.

Druk: Zakłady Graficzne w Katowicach. Zam. 0777/1333/7 K-95
Nakład 100 315 egz.



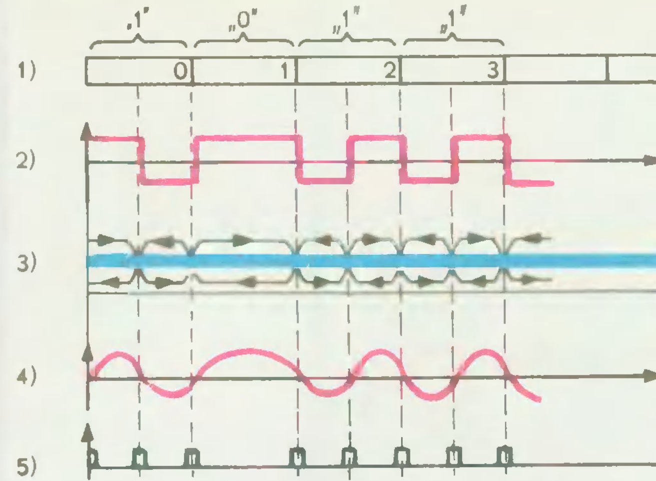
Dyskietka. 1 — koperta ochronna, 2 — wycięcie umożliwiające zapis, 3 — obszar nie używany ze względu na zbyt małą odległość od otworu mocującego, 4 — otwór wskazujący kontrolerowi pozycję początkowych sektorów na ścieżkach, 5 — ścieżka, 6 — otwór umożliwiający współpracę głowicy z nośnikiem magnetycznym



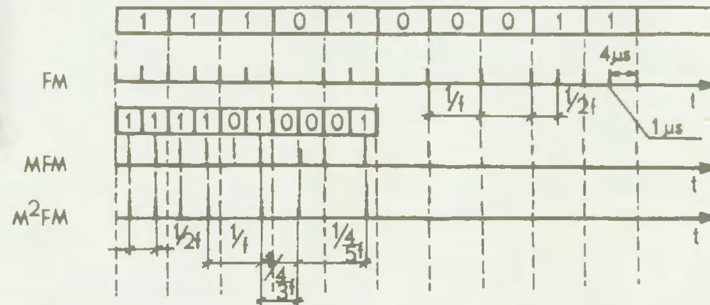
Symbole na opakowaniach dyskietek

Cena: 90 zł

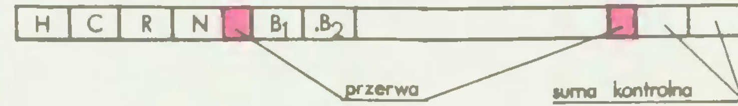
Indeks nr 366013



Rys. 1. Metoda zapisu sygnału na dyskietce: 1 — komórki z zapisaną informacją, 2 — prąd w uzwojeniu głowicy zapisującej, 3 — namagnesowania nośnika magnetycznego, 4 — napięcie w głowicy odczytujące, 5 — impulsy za komparatorem okienkowym



Rys. 2. Porównanie różnych rodzajów modulacji stosowanych do zapisu informacji



Rys. 3. Format sektora na dyskietce: H — numer głowicy, C — numer ścieżki, R — numer sektora, N — ilość bajtów wchodzących w skład rekordu, B_1, B_2 — bajty rekordu